Tampere University

Arjun Venkat Venkatakrishnan

Event Classification from Video Sequence Data

Faculty of Information Technology and Communication Sciences (ITC)
Master of Science thesis
September 2019

ABSTRACT

ARJUN VENKAT VENKATAKRISHNAN: Event Classification from Video Sequence Data
Tampere University
Master of Science thesis, 54 pages
September 2019
Master's Degree Programme in Information Technology
Major: Data Engineering and Machine Learning
Examiners: Dr. Pasi Pertilä, Dr. Mikko Parviainen
Keywords: Machine Learning, Neural Networks, Classification, Deep Learning, Action Recognition

---

The process of identifying a specific event from a video is a relatively easy task for humans. However, it is challenging for a machine to perform the same task. Deep neural networks, trained on huge datasets with adequate processing power, have achieved commendable results in image classification tasks. The natural extension is to use the existing approaches with added additional processing steps and solve the event classification from videos. In this thesis, three neural network based models Convolutional Neural Network (CNN), combination of a CNN with Long Short Term Memory neural network (CNN-LSTM) and three dimensional CNN (3D-CNN) are implemented to solve the task. In this thesis the implemented architectures are referred to as CNN, CNN-LSTM and 3D-CNN. Multiple experiments are performed to find out the most suitable model architecture. For CNN and 3D-CNN, the architectures are trained from scratch. In the case of CNN-LSTM, a pre-trained CNN model is used for extracting features. However, the LSTM architecture is trained from scratch.

The models are trained to identify specific events from two different video clip datasets. The UCF101 dataset, which is publicly available, is used as one of the datasets. This dataset consists of video clips of humans performing 101 different actions, four of which are examined here. The second dataset consists of video clips in which humans walk through a doorway. The events of interest in the second dataset are a person's entry and the exit. The data was collected from Tampere University premises and evaluated. The tested architectures are compared by estimating their classification accuracy. The additional factors involved in the architecture evaluation such as utilization of resources and memory consumption are also examined. Out of the three architectures, CNN-LSTM performs better than the other two ar-

chitectures. It produces best results, in terms of accuracy for both the datasets. CNN-LSTM achieves an average accuracy of 98.14 % for the UCF101 dataset and 98.9 % for detecting entries and exits. Additional experiments are also performed to determine the minimum amount of data required to reach certain accuracy. Using 50 % of the total data, CNN-LSTM was able to achieve 98.8 % accuracy.

PREFACE

First of all, I would like to thank Dr. Pasi Pertilä for his constant guidance, immense patience, and providing me with an opportunity to complete my thesis. I would like to thank Dr. Mikko Parviainen for answering my many queries and constantly giving me new ideas, with regards to implementation and technical writing. I would also like to thank Gaurav, Shanshan and Sowmya for giving valuable inputs and helping me to get used to a new environment.

Studying abroad in a completely new environment can be challenging, thanks to Nick for helping me with everything since the day we met. I would like to thank Rachayita for supporting me throughout this journey and never hesitating to give an honest opinion. Last but not the least, thanks to my parents, sister and friends for their constant support and faith in me.

Arjun Venkatakrishnan

Tampere, 9.9.2019

CONTENTS

1. Introduction 2

2. Overview of Basic Concepts 4

2.1 Machine Learning 4

2.2 Neural Networks 6

2.3 Convolutional Neural Networks 7

2.4 Recurrent Neural Networks 13

2.5 Performance Measures 15

2.6 Dataset Splitting 17

2.7 Related Research 17

3. Datasets 20

3.1 Door Dataset 20

3.2 UCF101 dataset 21

3.3 Rationale Behind Pre-Processing 23

3.4 Additional Data for Model Evaluation 23

4. Neural Network based Classification Models 27

4.1 CNN Model 27

4.2 CNN-LSTM Model 29

4.3 3D-CNN/C3D Model 32

5. Performance Evaluation of Implemented Models 35

5.1 CNN Model Performance 35

5.2 CNN-LSTM Model Performance 37

5.3 3D-CNN Model Performance 41

5.4 Rationale Behind Model Selection 43

6. Interpretations and Insights 47

6.1 Performance of CNN Model 47

6.2 Performance of CNN-LSTM Model 47

6.3 Performance of 3D-CNN Model 49

7. Conclusion 50

Bibliography 51

LIST OF FIGURES

2.1 Neuron Structure 7

2.2 Neural Network 8

2.3 Convolutional Neural Network 8

2.4 2D Convolutions 9

2.5 2D Max Pooling 10

2.6 ReLU 11

2.7 Sigmoid 12

2.8 RNN 14

2.9 LSTM 15

3.1 Entry Event 21

3.2 Exit Event 21

3.3 Makeup Event 22

3.4 Lipstick Event 22

3.5 Archery Event 22

3.6 Baby Crawling Event 23

3.7 Multiple Events-1 25

3.8 Multiple Events-2 25

3.9 Multiple Events-3 25

3.10 Multiple Entry Event 25

3.11 Additional Archery Event 26

3.12 Additional Archery Event 26

3.13 Additional Archery Event 26

3.14 Additional Archery Event 26

4.1 CNN Model Architecture 28

4.2 VGG-Feature Extraction 30

4.3 LSTM-architecture 31

4.4 3D-CNN Architecture 33

5.1 CNN—Training Performance Curves for the Door dataset (a) Accuracy versus Epochs (b) Loss versus Epochs 36

5.2 CNN—Training Performance Curves for the UCF101 dataset (a) Accuracy versus Epochs (b) Loss versus Epochs 36

5.3 CNN—Confusion Matrix (a) Door dataset (b) UCF101 dataset . . . 37

5.4 CNN-LSTM—Training Performance Curves for the Door dataset (a) Accuracy versus Epochs (b) Loss versus Epochs 38

5.5 CNN-LSTM—Training Performance Curves for the UCF101 dataset (a) Accuracy versus Epochs (b) Loss versus Epochs 38

5.6 CNN-LSTM—Confusion Matrix (a) Door dataset (b) UCF101 dataset 39

5.7 Data Required 40

5.8 3D-CNN—Training Performance Curves -Frames from Select Intervals for the Door dataset (a) Accuracy versus Epochs (b) Loss versus Epochs 41

5.9 3D-CNN—Training Performance Curves -Frames from Select Intervals for the UCF101 dataset (a) Accuracy versus Epochs (b) Loss versus Epochs 41

5.10 3D-CNN—Training Performance Curves -Frames Random Selection for the Door Dataset (a) Accuracy versus Epochs (b) Loss versus Epochs 42

5.11 3D-CNN—Confusion Matrix (a) Door dataset (b) UCF101 dataset (c) Door dataset-Random frames 43

5.12 CNN-LSTM—Additional Data Confusion Matrix (a) Door dataset (b) UCF101 dataset 45

5.13 3D-CNN —Additional Data—Confusion Matrix (a) Door dataset(Frames-Specific intervals) (b) Door dataset(Frames- Random selection) . . . 46

6.1 Misclassified Events 48

6.2 Misclassified Events-2 48

LIST OF TABLES

3.1 Number of videos per class in the Door dataset 21

3.2 Number of videos per class in the UCF dataset 23

3.3 Additional Videos in the Door dataset 24

3.4 Additional Videos in the UCF101 dataset 24

4.1 Number of Frames per split 29

5.1 Number of Frames per Dataset and Accuracy Metrics 37

5.2 Number of videos per Dataset and Accuracy Metrics 40

5.3 Amount of Data used and Accuracy 40

5.4 Number of videos per Dataset and Accuracy Metrics 43

5.5 Models on which additional testing is done 44

5.6 Additional videos per Dataset and Accuracy Metrics 45

5.7 Door Dataset Video Count and Accuracy Metrics 45

LIST OF ABBREVIATIONS AND SYMBOLS

CNN	Convolutional Neural Networks
GPU	Graphics Processing Unit
LSTM	Long Short Term Memory
OpenCV	Open Source Computer Vision
RNN	Recurrent Neural Networks
SGD	Stochastic Gradient Descent
URL	Uniform Resource Locator

1. INTRODUCTION

An event can be considered as an action or a set of actions occurring within a time frame, performed to complete a given task. The time frame can be of varying length. These actions are performed by humans and there might be other objects involved, while performing the event. An event can be complete or incomplete with different actions involved. For example, a door not closed, single person performing two different events and multiple people performing the same event. The process of identifying specific events from a given sequence of videos is known as an event classification problem.

The task of identifying specific events from a video involves understanding and processing subtle changes in consecutive video frames. For humans, performing this task is effortless. To replicate the same on a machine is an interesting task to solve. In the field of automated image classification, impressive results have been achieved. Recent advancements in utilization of neural networks are one of the reasons for better results. Datasets such as ImageNet [9] used in conjunction with state-of-the-art neural networks have produced results with high accuracy in the image classification task. Since a video is a collection of consecutive frames, it is possible to approach the problem of video classification as an image classification problem with enhancements for handling temporal dependencies.

The goal of this thesis is to design neural networks based models that can perform the task of event classification on video data. Different neural network architectures are tested and their performance is evaluated to choose the best model for the event classification task. Three neural network based models are designed in this thesis namely, Convolutional Neural Network (CNN), three dimensional CNN (3D-CNN) and combination of CNN with Long Short Term Memory (LSTM) neural network. Two datasets are used to train these models. One of the datasets is publicly available, UCF101 [30]. The other, custom dataset consists of videos of people walking through a doorway, gathered at Tampere University premises for thesis work.

Image classification techniques cannot be directly applied to event classification from

videos. Additional temporal processing is required such as using a combination of neural networks (CNN-LSTM) to solve the task. The spatial information across multiple frames are captured by CNN and passed on to LSTM. LSTM handles the subtle changes in the frames with respect to time. Once desired results are achieved by the models, experiments are concluded.

This thesis follows the structure of chapters and sections. Each chapter indicates a particular stage in the thesis and the chapters consist of multiple sections and sub-sections. Multiple images, graphs and tables are used to provide an insight into the performance and results achieved. Basic overview about various concepts used in the thesis are introduced in Chapter 2. Detailed information about the datasets used and their pre-processing is explained in Chapter 3. Once the basic understanding of the thesis work is established, neural networks based classification models are introduced and explained in detail in Chapter 4. Model architectures and pre-processing required for each model is explained. Chapter 5 focuses on the performance of the models in both training and evaluation phases. Furthermore, additional testing performed on the models is also covered. In Chapter 6, the performance of the models is analyzed and compared. Finally, the thesis concludes by choosing one of the three implemented models for further experiments.

2. OVERVIEW OF BASIC CONCEPTS

This chapter focuses on various concepts used in the thesis. A brief description about machine learning, deep learning, neural networks, and performance metrics is discussed. It concludes by presenting three models for event recognition.

2.1 Machine Learning

Machine learning is a branch of artificial intelligence, which refers to machines (e.g. computer program) performing with a certain level of intelligence. Machine learning is an approach by which a computer program solves a particular task, based on various examples or data provided. Moreover, instead of explicit programming, the program gets trained from the data [6].

A machine learning program can be designed to understand trends, interesting attributes from the data provided. The program is expected to take decisions based on the data on which it was trained. During the training process, the tasks performed by the machine are constantly evaluated. Given that the training is successful, the tasks performed by the program improves significantly from previous experiences [23]. Some of the well known machine learning applications include image classification [4], speech recognition [10], and recommendation systems [19].

2.1.1 Learning Techniques

There are different approaches that are used to make a computer program to learn. The most common ones are supervised learning and unsupervised learning.

- **Supervised Learning**

In the supervised learning approach, the computer program works with the input data and the desired output response for each input data sample. The output responses are called labels for classification tasks. Pairs of input and output are

provided to the program [28]. After training, the program is evaluated by testing it with the data that is yet to be seen or unseen. The output responses on unseen data made by the program are known as predictions. If the prediction made by the program corresponds to the label, the prediction is considered to be correct. The number of those correct predictions made by the program is essential to evaluate its performance.

Classic examples of supervised learning are classification and regression tasks. Classification, as the name suggests, involves classifying the input data to corresponding labels. The program is given examples of data and corresponding labels in the form of input and output pairs. For example, filtering spam emails fall under the category of classification. The task of event classification from videos, falls under the category of supervised learning and it is a classification task.

On the other hand, in regression, the computer program acts as an estimation function. The input to the regression model can be either scalar or vector. The output could be a scalar, a vector, a matrix or a tensor. Forecasting the price of a particular product, curve fitting are some of the examples of regression tasks.

- **Unsupervised Learning**

In the unsupervised learning approach, the computer program works with unlabeled data. The program is expected to analyze the data provided and make its own interpretations, hence the name unsupervised learning. Classic examples of unsupervised learning include clustering and association tasks. Clustering involves grouping the data based on the similarities or commonalities, whereas in association the program tries to establish a relationship between the variables of data provided.

K-means clustering, one of the clustering algorithms, involves in identifying the clusters present in the data provided. The goal of this algorithm is to have minimum distance between the data belonging to same clusters and maximum distance between the clusters [2]. The Apriori algorithm [1] is one of the examples of association. This algorithm aims to generate rules or associations from the provided data. Different variations of Apriori algorithm and K-means clustering exist.

- **Other learning methods**

Semi-supervised learning [39] and reinforcement learning [32] are the remaining learning techniques. These learning techniques are outside the scope of this thesis.

2.1.2 Deep Learning

Deep learning falls under the umbrella of machine learning and it is used to solve complex problems. The term deep refers to using neural networks with multiple layers. Supervised machine learning algorithms require a sufficient amount of labeled data. If the task of the algorithm is to classify images, it is advisable to have images captured with different backgrounds and camera positions. This technique will improve the model's accuracy and make it robust. From the data, the program learns and performs the required tasks. However, it is not the easy to determine which factors or features to look for when performing the task [14].

The problem of feature selection becomes more relevant in the case of classifying images. It is hard to identify characterizing attributes of an image, in the form of pixel values. This task is solved by deep learning techniques, which involves understanding representations (features) from the given data [14].

2.2 Neural Networks

Neural networks or artificial neural networks mimic the biological neural networks. A neural network is made up of multiple units which are interconnected. There is a flow of information, as well as information storage in the smaller units. Neural networks can be trained to execute a given task, based on a well-defined learning process [16]. The ability to learn and store the learned information makes them suitable for performing complex classification tasks. Individual units of an artificial neural network are called as artificial neurons.

Typical neural network architectures include fully-connected feed forward neural networks, convolutional neural networks, and recurrent neural networks. In feed forward neural networks and convolutional neural networks, the flow of information is only in one direction. However, in recurrent neural networks there exists a loop or loops in the flow of information.

- **Structure of an Artificial Neuron**

A visual representation of an artificial neuron is shown in Figure 2.1 adopted from [16]. The important parts of neuron to be considered are the inputs $(x_1, x_2, \ldots, x_n)$ multiplied with corresponding weights $(w_1, w_2, \ldots, w_n)$ and the connections are referred to as synapses. The values of the weights can be either positive or negative. A constant-valued bias is added, which may impact the activation values.

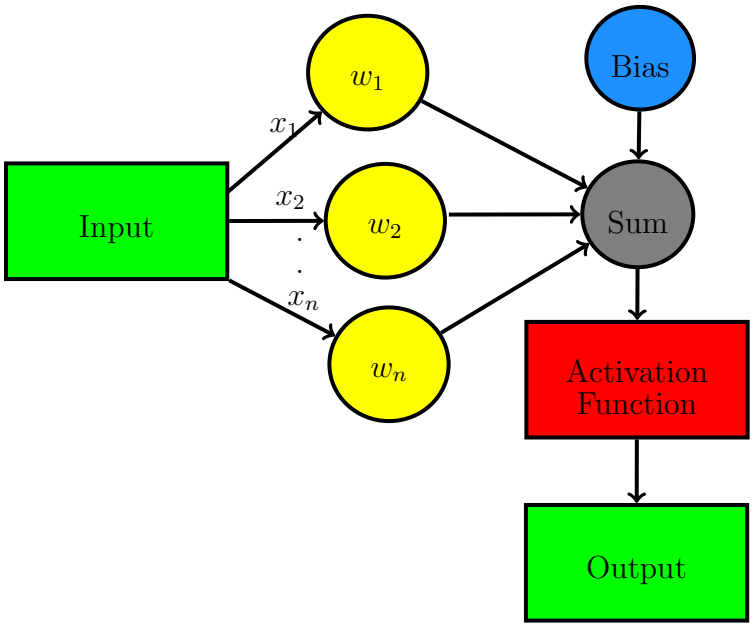


Figure 2.1 *Structure of a neuron. The input is in the form of a vector. There exists a mapping between the inputs and the output.*

The inputs along with the corresponding weights are added using summation process. The activation function is used to set the range of the output values, some activation functions exhibit a squashing effect on the values [16]. Sigmoid, threshold, and Rectified Linear unit (ReLu) are some of the activation functions.

- **Layers of a Typical Neural Network**

A neural network minimally consists of three layers, namely input, hidden and output layers. There can be multiple interconnected hidden layers, which act as the connection between input and output layers. It works on the weighted inputs from the input layer, transforms it and sends them to the activation function. Neurons of the hidden layer gets values from all input weighted neurons or from the outputs of the previous hidden layer. A typical structure of a feed forward neural network is shown in Figure 2.2.

2.3 Convolutional Neural Networks

A neural network is said to be a deep neural network, if there are multiple layers. Convolutional Neural Networks or CNN are a category of deep neural networks. They perform an operation called convolution on the input data and hence the name convolutional neural networks. These networks are ideal for working with images or

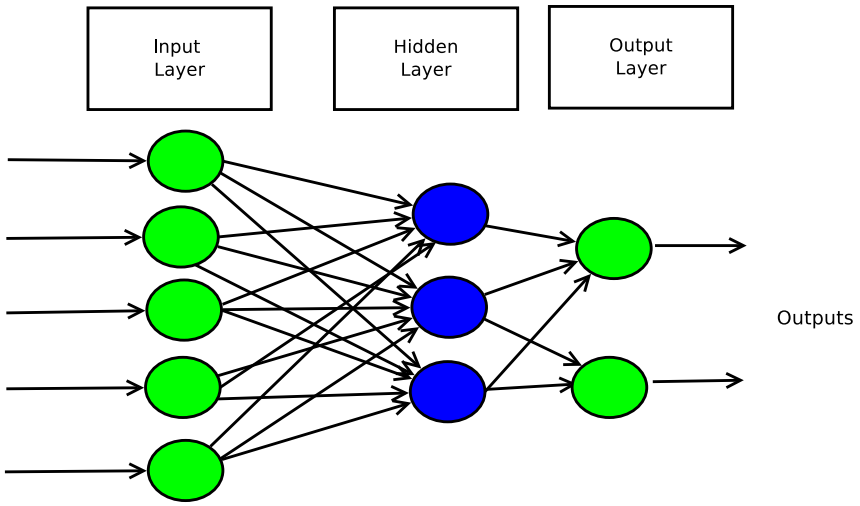


Figure 2.2 *Structure of a typical Neural Network*

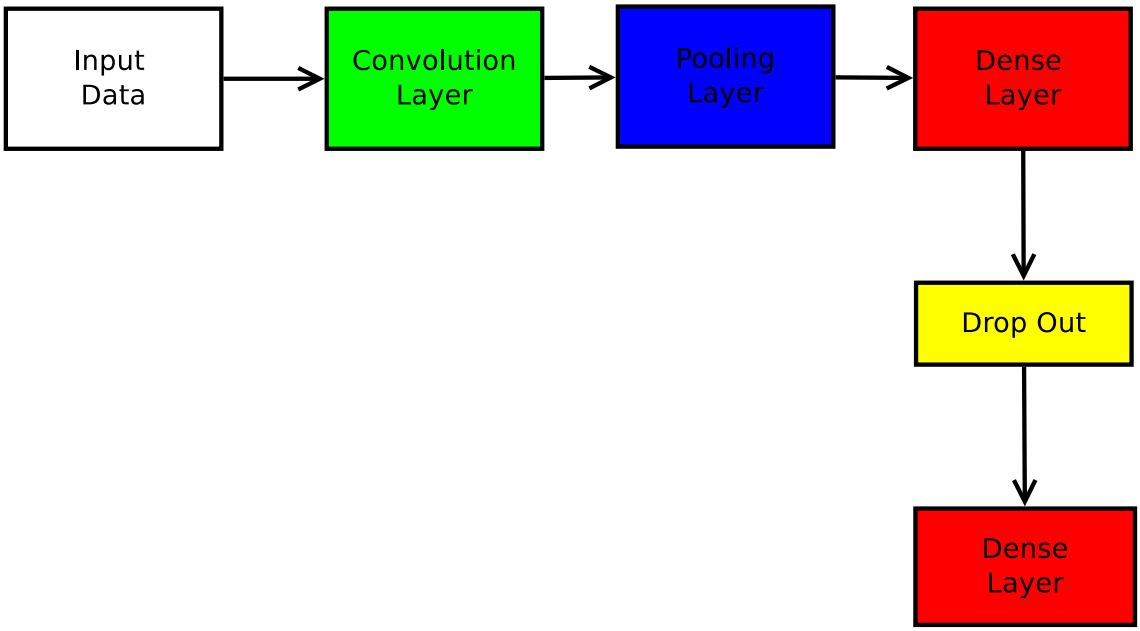


Figure 2.3 *Structure of a typical Convolutional Neural Network*

visual data, due to their ability to detect objects irrespective of their position in an image i.e. transitional invariance. A typical convolutional neural network consists of layers such as convolution layer, pooling layer, drop out layers, and activation. Figure 2.3 depicts the block diagram of a CNN.

- **Convolution Layer**

The input data usually images, is read in the form of pixels. The data, which is read is represented as (height, width, depth). Depth refers to the number of channels,

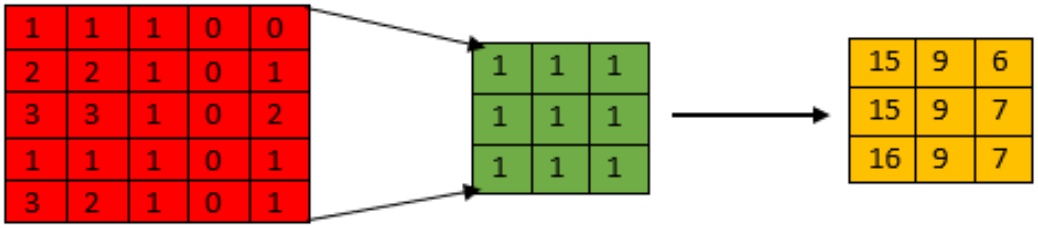


Figure 2.4 *Filter of size (3×3) is moved across the (5×5) matrix. Pairwise multiplication followed by addition is performed.*

denoting whether the image is a color image or gray-scale image. The data is read in the form of a multidimensional matrix. A filter or kernel is used, such that the depth of the images and the filter are usually of the same value. Convolutions [22], i.e. element-wise multiplication, followed by summation are performed between the input data and the filter values. In this thesis, '×' denotes the kernel size and '.' denotes the regular multiplication.

The process of convolution is depicted in Figure 2.4. As shown in the figure, element-wise multiplication is performed between the pixel values matrix and the filter matrix. The filter is moved over the pixel values matrix during the process of convolution. This process is repeated until all pixels are covered and it leads to the generation of feature maps. These feature maps are multidimensional arrays. The size of the kernel, and the number of feature maps are chosen by the designer of the network. In 2-D convolutional neural network, a 3-D filter is moved over 3-D images. Since the depth of the filter and depth of the image is the same, the convolutions are performed only in two dimensions (height and width), hence the output is 2-D in nature.

Stride- It is the number of pixels the filter will move, after the filter has processed a set of pixels. If the values of stride is two, the filter will move two pixels at a time.

Padding- If the size of the filter and size of the image/pixel value matrix does not match, the image is padded with zeros. This process is called zero padding and it allows the size of the output to match the size required or input image size. Another approach is to ignore the part of the image that does not match with the filter, which is called valid padding.

- **Pooling Layer**

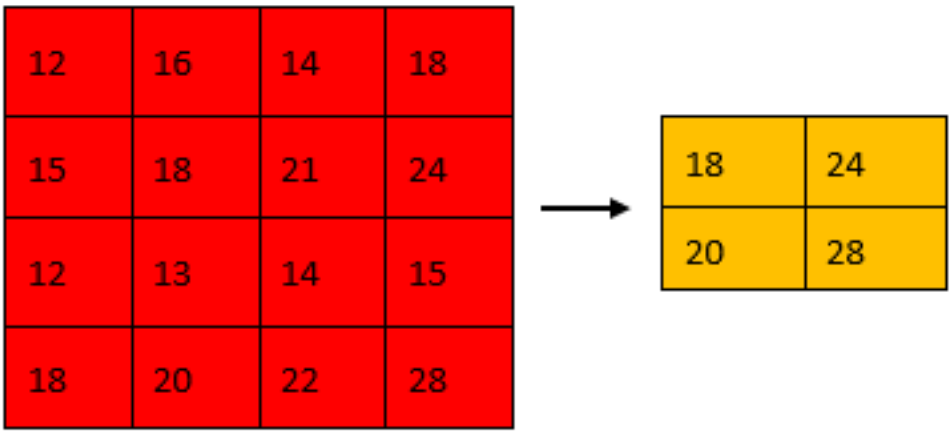


Figure 2.5 *Max pooling: Maximum value from each* (2×2) *position is fetched resulting in a final matrix on the right.*

In order to extract relevant features from an image and to achieve positional invariance [12], a process called pooling is conducted. The values obtained from the activation function (after convolution) are passed on to the pooling layer. Pooling reduces the network parameters and also the size of the image (feature map). However, depth in the pooling process is retained. In pooling layers, arithmetic operations or aggregation [38] are performed. The common approaches of aggregation include choosing the maximum value among pixels known as max pooling. Another approach that involves performing average over all the values, known as average pooling.

However, these pooling methods may not yield the best results and might lead to overfitting (refer Section 2.5.1) of the model. [12] and [38] discuss new approaches of performing pooling operations, which involve calculating geometric mean, harmonic mean, or approaching it as a probability distribution. The factor by which the input data should be reduced or downsized is specified by the neural network designer. A visual depiction of the pooling operation is shown in Figure 2.5.

- **Dense Layer**

After series of convolutions and pooling, the extracted features are passed into fully connected layers or dense layers. Linear operation is performed on the data, such that every input is connected to every output. In addition to that, an activation

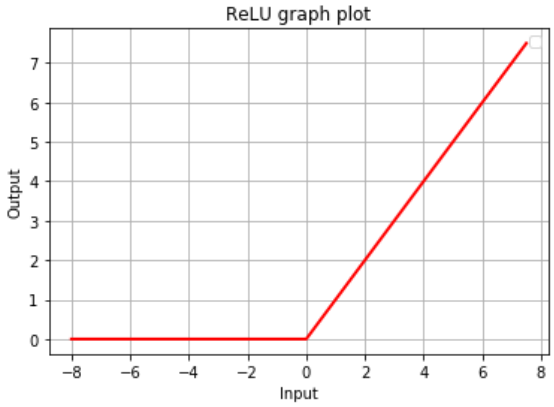


Figure 2.6 *ReLU graph*

function is applied on the data. Before passing the data to the dense layer, the data is flattened, i.e. vectorized using a flatten layer.

The dense layer also acts as the output layer. The number of parameters or units in the final dense layer is same as the number of output classes, if it is a classification problem.

- **Activation functions**

After the process of convolution, the output values are passed through an activation function. The activation functions which are used in the thesis along with dense layers are Soft Max and Sigmoid. In terms of convolution layer, ReLU (Rectified Linear Unit) is used as the activation function.

ReLU Activation

In ReLU [24] activation, if a particular value is greater than zero, it is retained else the value is set to zero (2.31). In other words, its output value is linear for positive input values and zero for negative input values. [37] compares and studies different variants of ReLU.

$$\text{ReLU}(x) = \begin{cases} x, & \text{if } x \geq 0 \\ 0, & \text{otherwise} \end{cases} \tag{2.31}$$

Figure 2.6, plots the operation of ReLU activation.

Sigmoid Activation

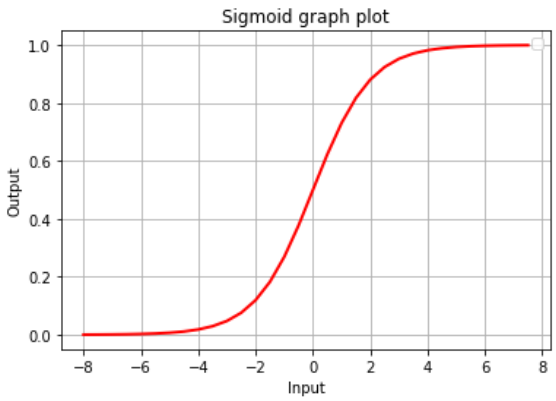


Figure 2.7 *Sigmoid graph*

The sigmoid activation produces output values in the range [0,1]. The formula for calculating sigmoid activation is shown in (2.32). Visual depiction of sigmoid curve is shown in Figure 2.7. A smooth curve is achieved and there is some output value, irrespective of the sign of inputs.

$$\text{Sigmoid}(x) = \frac{1}{1 + e^{-x}} \tag{2.32}$$

SoftMax Activation

The output of SoftMax activation is the probability distribution among the number of units in the final dense layer, which makes it suitable for multiclass classification. The formula for calculating SoftMax activation is represented in (2.33) [34], where i denotes class index or output unit index and j is the number of classes or output units. To obtain individual output instead of the probability distribution among outputs, argmax can be applied to the result of SoftMax activation, which in turn is the predicted class.

$$\text{SoftMax}(y_i) = \frac{e^{y_i}}{\sum_i^j e^{y_j}} \tag{2.33}$$

- **Training of a Neural Network**

The input data is passed through multiple layers of a neural network. Once it reaches the end, the neural network performs prediction on the input data. A cost function calculates the discrepancy between the expected and actual results produced by the neural network. Cross entropy is the loss function used while training the neural

network for the classification task. In this scenario, there are two types of loss functions used namely binary cross entropy and categorical cross entropy. When there are multiple classes to be classified, categorical cross entropy is used as the loss function. Categorical accuracy is used as the performance metric and it compares the index of predicted value and index of original (true) value. If the values are the same, the prediction is considered as correct. Mean Squared Error (MSE) is one of the cost functions used for regression tasks. The cost is the difference between the model's output and the expected output, squared and averaged over all the cases. Ideally, the cost should be as low as possible.

The difference or error is sent backwards through the network which is called backpropagation. The main idea behind backpropagation is to recursively solve the derivatives of the weights and biases with respect to the error. The changes are backpropagated from the output layer to the current layer. A neural network features several parameters, including batch size and learning rate. These parameters, called hyperparameters, define the network architecture. Hyperparameters are initialized before the training phase of the neural networks. Hyperparamter adjustments are required to find out the suitable neural network architecture for the task. Modifications are required for the weights and biases as well.

Instead of manually changing the weights, the gradient descent algorithm can be applied. The aim of the gradient descent is to change the values of the weights such that the error is reduced [27]. In other words, it is an optimization algorithm.

This thesis uses Stochastic Gradient Descent (SGD), which performs parameter update for each training data sample encountered [27]. Frequent changes made by SGD might result in high variance i.e. fluctuations in the error. A critical value in SGD is the learning rate, which defines how much weight values are changed with each update. The learning rate of the algorithm is a delicate parameter and it should be therefore carefully chosen and adjusted by observing the progress of the training. With a poorly chosen learning rate value, the training is either impractically slow or divergence occurs i.e. training and validation curves diverge.

2.4 Recurrent Neural Networks

In recurrent neural networks, the output not only depends on the current input, but also previous inputs. It works similar to a feedback system. Some of the applications of RNN include speech recognition [10]. Typical structure of a RNN is depicted in Figure 2.8. It is similar to a fully-connected feed forward or dense network, except for the loops, which indicates flow of information in two directions.

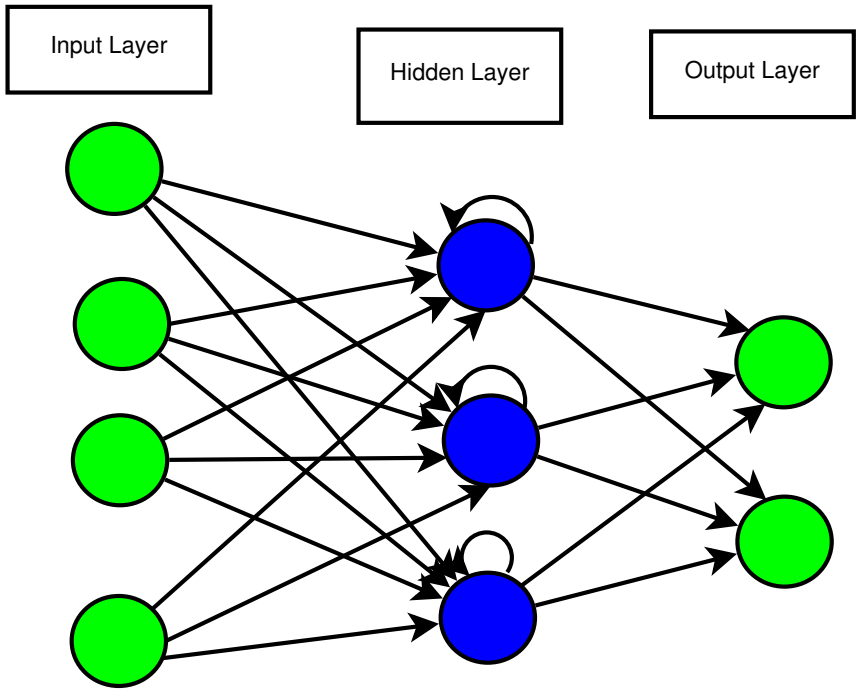


Figure 2.8 *Typical structure of a RNN. The loops indicate the flow of information.*

Similar to feed forward networks, RNNs are trained using the backpropagation algorithm (see Section 2.3). Since the output of the current layer in RNNs depends on multiple previous layers, the weights might be adjusted several times for the same neuron. Training of RNNs can be difficult due to the problems known as vanishing and exploding gradients [18].

2.4.1 LSTM

The LSTM [18] is a certain type of RNN. LSTM neural networks address the problem of gradients' behaviour with the help of memory cells. The flow of information through each LSTM unit can be controlled and its memory cells enable remembering input sequences.

A typical LSTM cell consists of three gates called an input gate, a forget gate, and an output gate. The forget gate decides which information will be stored and which will be erased. The input gate makes a decision on the information which will be saved, combined with the data from forget gate. The decision on the output of the cell is made by the output gate. Filtered data from the previous cell state is also considered [7]. Figure 2.9 provides a block diagram of a typical LSTM cell.

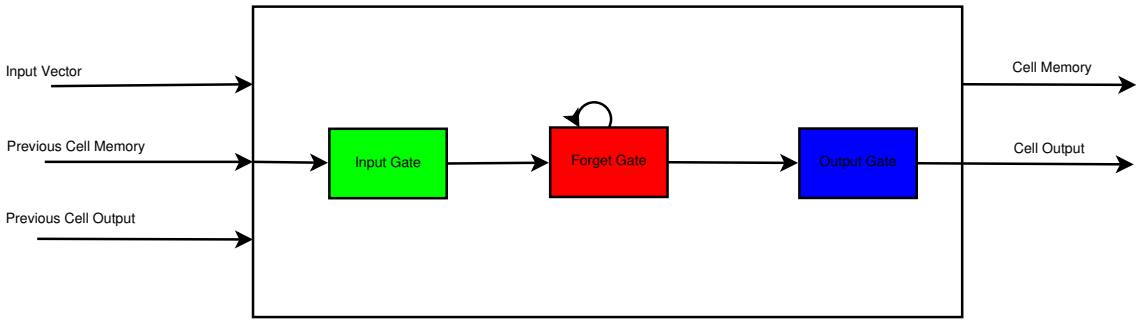


Figure 2.9 *LSTM block diagram*

2.5 Performance Measures

The final accuracy is calculated using the formula (2.34).

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} \tag{2.34}$$

In order to get insights into model's performance, additional performance metric, f1-score is used in this thesis. To calculate f1-score, two other performance metrics, namely precision and recall are required. The precision metric is the fraction of the relevant predictions to the retrieved predictions, whereas the recall metric focuses on the correctness of the relevant predictions. These performance measures are useful when data imbalance between classes exists i.e. more data is available for one class compared to other class or classes. The efficiency of a classifier can be estimated using the above-mentioned metrics.

The terms used in calculating precision and recall are True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN).

Assume predictions are performed on a specific object and there are two classes namely positive and negative.

- TP and TN indicate that the model has predicted the correct class for the object under consideration. The objects are classified to their respective classes.

- FN and FP indicate the presence of an incorrect classification. Objects belonging to positive class are classified as the negative class and vice-versa.

Precision is calculated using the formula (2.35)

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2.35}$$

Recall is calculated using the formula (2.36)

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2.36}$$

The precision metric can be interpreted as probability of predicting a relevant or correct prediction [15], while the recall metric is the probability of correctness of the predictions made [15]. Higher values of precision and recall indicate that the model is performing well. Conversely, if one of the values is higher and the other one is lower, it indicates that the model is unable to perform classification correctly.

f1-score is the harmonic mean of precision and recall. It is computed using the formula shown in (2.37)

$$\text{f1-score} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision+recall}} \tag{2.37}$$

In this thesis, inbuilt f1-score function of scikit-learn [26] is used. The function requires predictions made by the model and the ground truth or original labels to calculate f1-score. The output is in the form of an array and each class has its own value. Ideally a model that performs well should have f1-score values close to 1.

So called confusion matrix is generated to depict pictorial representation of classifications made by the model. The number of ground truth values per each class is specified in the rows. Actual predictions made by the model are represented in the columns. If there are zero incorrect classifications, only the diagonal elements are filled in the confusion matrix. The off-diagonal elements represent the number of incorrect classifications made and to which class.

2.5.1 Overfitting and Mitigation Techniques

Overfitting occurs when the trained model performs and achieves good accuracy with the training data, but its the accuracy is relatively low with the test data. In order to avoid overfitting, a technique known as dropout is used. In this technique, some of the neurons or units are dropped and this process is completely random [31]. Dropouts are implemented by multiplying weight values with zero. The neurons

along with the connections are dropped from the neural network during training, and no dropping but scaling during testing. Instead of adjusting weights of all the neurons, especially in large neural networks, the model will learn from a reasonable number of neurons [31].

In addition to dropouts, the early stopping technique is also used to avoid overfitting. In this thesis, validation loss is used to determine when the training needs to be stopped. The process of feeding the dataset once through the model is known as an epoch. After each epoch or after a mini-batch of samples, the model parameters i.e. weights and biases are adjusted based on the performance of the model. An arbitrary value is set for number of epochs to be considered. The training stops, if the validation loss does not improve within the specific number of epochs. The training instance with the least validation loss value is saved.

2.6 Dataset Splitting

The dataset is usually split into three categories: training, validation, and test set. A typical split could be that 80 % of data is used for training and validation and the remainder is used for testing (unseen data). However, the ratios vary depending on the data availability. The dataset should be able to reflect all possible variations in real-life conditions. Moreover, the same data should not be used both for training and testing the model. The model gets trained on the training set and the performance of the model is evaluated using the validation set. Based on the performance on the validation set, the model parameters are adjusted using the backpropagation algorithm. Once the entire training process is completed, the model is evaluated again using the testing set. However, the model parameters are not adjusted or changed in evaluation with testing data set. The performance of the model on the test data set is the ultimate score of the model and is used when comparing it to other models.

2.7 Related Research

This section focuses on existing neural network based approaches for the event classification task. Though multiple approaches exist, three approaches can namely, CNN [20] based approach, 3D-CNN [35] based approach and CNN-LSTM [25] based approach can be considered as primitive approaches. Advanced methods like Two Stream Fusion [13], Temporal Segmentation Networks [36] and Long term Recurrent Convolutional Networks [11] are built by modifying and enhancing the primitive approaches. However, the amount of time and resources required for pre-processing

and training the models might be significantly higher than that of previously mentioned three approaches.

• **CNN-based approach**

Since a video is a series of images (frames), image classification is performed on the individual frames. As discussed in Section 2.3, CNNs are suitable for working with image classification tasks. In [20], the videos were split into frames and the CNN model was trained on those images. The CNN model was trained on UCF101 [30], additional details about the dataset is described in Section 4.2. The accuracy of using this CNN based model for event detection was around 41 %, for frame-by-frame case. The process of sending one frame after the other yielded relatively low accuracy. One of the main reasons was the models inability to exploit the temporal dependency of the data.

• **3D-CNN-based approach**

The problem of handling temporal data was addressed by 3D-CNN [35]. 3D-CNN completely relies on 3D convolutions, which enables exploitation of both spatial and temporal properties. The video clips were cut such that each input to the model had 16 images (frames). There was an overlap of eight frames between two successive clips and the input to the model was of uniform length. The trained 3D-CNN model was used as a feature extractor [35] i.e. extracting characterizing features from the video clips. The extracted features were provided as input to a multiclass Support vector Machine (SVM). The predictions were done by the SVM, which aimed to maximize the difference between different classes [8]. The implemented 3D-CNN [35] achieved 85.2 % accuracy.

• **CNN-LSTM-based approach**

CNN-LSTM [25] method involves a pre-trained CNN model, to extract features. The features are then passed to a series of LSTM layers. An entire video is sent to the CNN-LSTM model and it predicts the event from the input video. State-of-the-art pre-trained models such as AlexNet [21] and GoogleNet [33] were used to extract the features from input video. UCF101 was used for training and evaluating the model. The extracted features were sent to the stack of LSTM cells, frame-wise. The predictions were then performed on each frame and then predictions were made

on the frames. On UCF101, on raw input frames and optical flow frames, achieved 88 % accuracy.

Based on the analysis, the CNN-LSTM approach achieved the best result. However, additional processing is required, such as generating optical flow images for achieving the specified accuracy. These images help in identifying the motion of an object or objects present in consecutive frames. 3D-CNN and CNN-LSTM, clearly outperformed CNN, in terms of accuracy.

3. DATASETS

This chapter briefs about the datasets, which are used to train the models. Two datasets are used for training and evaluating the models. Data collection and data pre-processing steps are also discussed here.

3.1 Door Dataset

Door Dataset or custom dataset consists of people entering and exiting through a door in a particular space. This data is collected from Tampere University premises. The data is captured using a Raspberry pi RGB camera, equipped with a software capable of detecting human motion. Once the motion is detected, the camera starts capturing a series of images. The captured data is concatenated into a single big video. The time-stamps at which the door is opened is annotated by hand using ELAN[†]. The time-stamps points to occurrence of an event it can be either Entry or Exit, since both involves opening the door.

The series of action involved in the Entry and Exit events are displayed in Figures 3.1 and 3.2. The events are very similar in nature, except for the direction of the occurrence.

3.1.1 Door Dataset Pre-Processing

A MATLAB[††] script is used to perform pre-processing on the dataset. The entire video is split into frames. Each event occurrence is indexed. The frame rate is 25 frames per second. The index position reveals the frame number where the event occurs. A complete event is initially marked to span 25 frames before and after the indexed frame. The script uses the annotation to extract appropriate frames from the master video.

The user can change the number of frames taken into consideration. Based on the event, both the selected frames and video created by merging frames are saved to

[†]https://tla.mpi.nl/tools/tla-tools/elan/
[††]www.mathworks.com

Figure 3.1 *An image sequence representing the Entry event. The person opens the door, enters and closes the door.*

Figure 3.2 *An image sequence representing the Exit event. The person walks through the door, exits and closes it.*

a folder, corresponding to the event type. The number of frames chosen and video duration can be of varying length. The frames that depict the event perfectly are categorized and saved based on the class label.

Since the task is to classify single Entry and Exit events, frames with multiple people and other anomalies are saved separately. Some of the videos containing other anomalies are used for performing additional testing on the models.

Table 3.1 *Number of videos per class in the Door dataset*

Event Name	**Videos**
Entry	315
Exit	132

As shown in Table 3.1, there is an unequal distribution of videos among the events. This issue is handled by splitting the training set so that there is equal distribution between the events.

3.2 UCF101 dataset

UCF101 [30] consists of videos belonging to 101 different human actions. Each action is regarded as a separate event. The videos were collected from YouTube and each video clip has a varying duration. A subset of events i.e. four events were taken into consideration. The four events are Apply Lipstick, Apply Eye Makeup, Archery, and Baby Crawling.

The requirement of having a common model architecture for both the datasets is one of the reasons to choose a subset of events from UCF101 [30]. Figures 3.3 – 3.6 illustrate each of the four events selected from UCF101 [30]. A short description of each event is specified below the images.

The dataset consists of videos captured from different camera positions and different backgrounds. A single video of long duration is split into smaller videos of varying duration. Except for Baby Crawling, all other events involve human-object interaction. Moreover, such interacting actions are repetitive in nature.

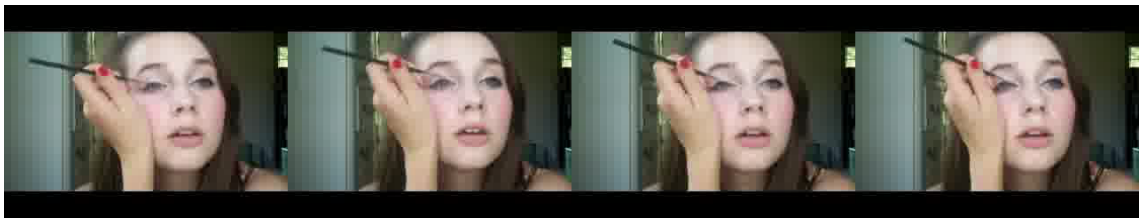


Figure 3.3 *This image series represents the Apply Eye Makeup event. The person is applying eye makeup and it is a repetitive action.*

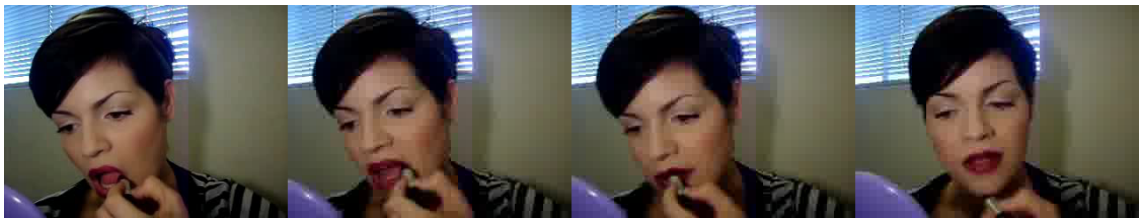


Figure 3.4 *This image series represents the Apply Lipstick event. The person is applying lipstick and it is a repetitive action.*

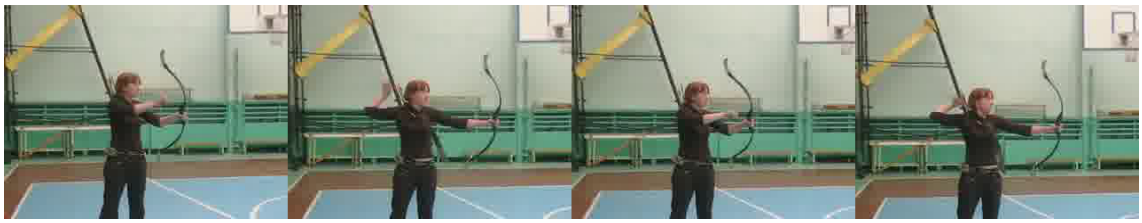


Figure 3.5 *This image series represents the Archery event. The person is drawing arrows from behind their back and hitting the arrows using a bow.*

3.2.1 UCF101 Dataset Pre-Processing

Since the dataset is already split into smaller videos, the pre-processing task involves converting the videos into frames. This is done by using FFMPEG* and the OpenCV

*https://ffmpeg.org/

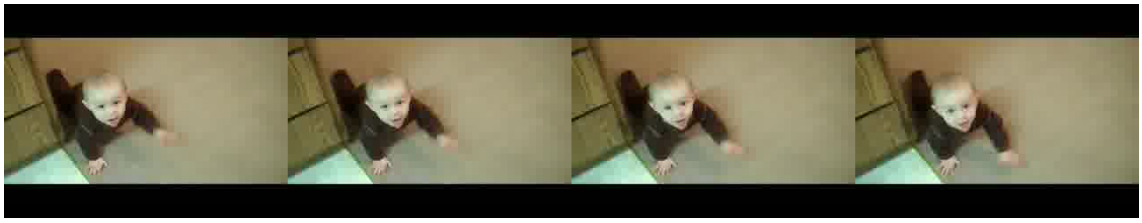


Figure 3.6 *This image series represents the Baby Crawling event. The baby just crawls on a surface. This action does not involve any interaction with other objects.*

Table 3.2 *Number of videos per class in the UCF dataset*

Class Name	Videos
Apply Eye MakeUp	145
Apply Lipstick	114
Archery	145
Baby Crawling	132

[3] Python library. Each video is processed and converted into series of images. It is possible for the user to set the rate at which the frames will be extracted from the videos. Since the number of videos per event is almost the same, there is no data imbalance issue. The total number of videos per each class is shown in Table 3.2.

3.3 Rationale Behind Pre-Processing

The implemented models can accept input as either videos or frames. Hence the datasets are pre-processed to both frames and videos. It is advantageous to have inputs in the form of frames, since it is easier to remove unnecessary or redundant frames if required.

If the input data is in the form of videos, OpenCV [3] can be used to extract frames from the videos. It is possible to extract frames at specific time stamps or all the frames from a video can be extracted.

3.4 Additional Data for Model Evaluation

In order verify the robustness of the models, additional data is used for testing them. The data includes incomplete events, videos with missing frames and additional data from YouTube (for the UCF101 dataset).

Table 3.3 *Additional Videos in the Door dataset*

Event	Videos
Entry	92
Exit	38

Table 3.4 *Additional Videos in the UCF101 dataset*

Event	Videos
Apply Lipstick	4
Apply Eye Makeup	4
Archery	3
Baby Crawling	4

3.4.1 Additional Data – Door dataset

In the case of the Door dataset, multiple people doing the same event is used as a part of the dataset. Test scenarios include cases where the same person performs two events, as well as missing frames and incomplete events such as leaving the door open. In addition to that, videos with complete events are also added. Visual examples with short textual description are provided in Figures 3.7 — 3.10.

The number of videos per event in the Door dataset is represented in Table 3.3.

3.4.2 Additional Data - UCF101

In the case of UCF101, additional videos were chosen from YouTube. The difference is the length of the videos and additional content present on each video, such as extra graphics and text. The quality of videos are comparatively better than that of UCF101's videos. The videos chosen are very similar to events considered in UCF101 dataset. Figure 3.11 — 3.14, represents few sample frames from these additional videos.

The number of additional videos per event in the UCF101 dataset, is shown in Table 3.4.

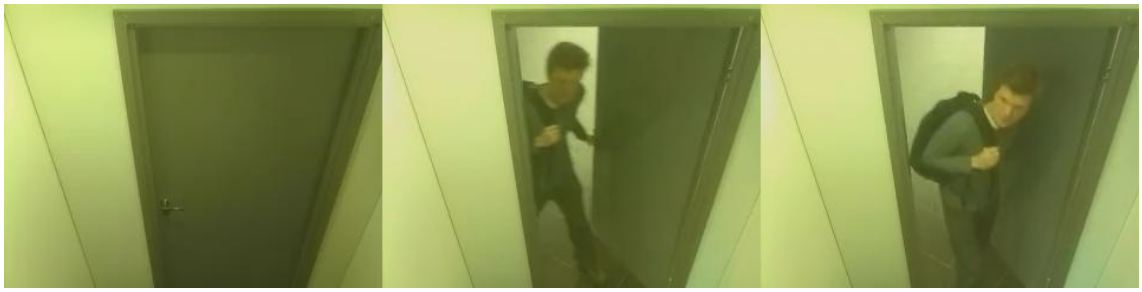


Figure 3.7 *This series of images represent the Entry event. A person walks through the door.*

Figure 3.8 *The Entry event is continued by the same person. However, the event is incomplete.*

Figure 3.9 *The same person begins the Exit event. Two events are performed by the same person consecutively.*

Figure 3.10 *This series of images represent the Exit event being performed by multiple people.*

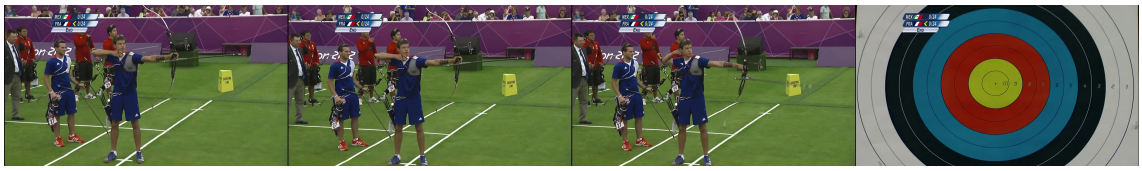


Figure 3.11 *This series of images represent the Archery event performed in the Olympics.*

Figure 3.12 *This series of images represent the Apply Lipstick event enacted as a tutorial video.*

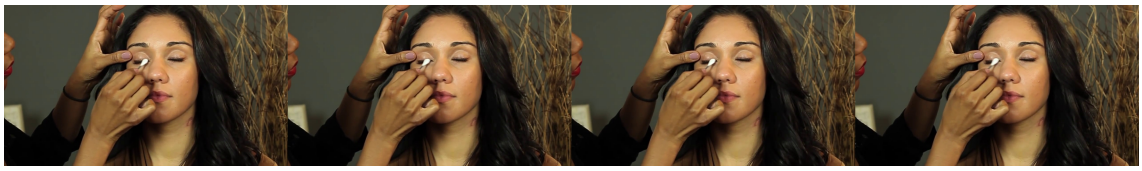


Figure 3.13 *This series of images represent the Apply Eye Makeup event performed as a tutorial video.*

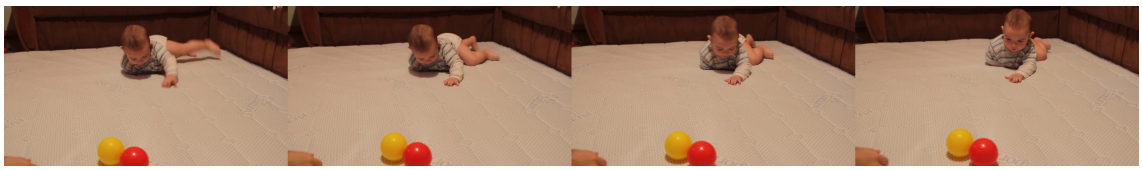


Figure 3.14 *This series of images represent the Baby Crawling event.*

4. NEURAL NETWORK BASED CLASSIFICATION MODELS

This chapter focuses on neural networks based model implementation. The details of model architecture and input pre-processing techniques are discussed for each implemented model.

In this thesis, two types of features are handled. Spatial features, are extracted from images. Spatial features includes identifying edges, certain characterizing shapes from images. The features which change with time are called temporal features. These features aid in capturing the slight variation between the frames. Hence temporal features play a vital role in event classification tasks focused on this thesis.

4.1 CNN Model

The approach behind this model can be considered as naive one, due to the relative simplicity of its implementation. The idea is to segregate all frames belonging to a particular event and feed frame by frame as input to the classification model. The model learns the spatial features from the individual frames.

4.1.1 CNN Architecture

Since the idea is to have one common architecture for both the UCF101 and the Door datasets, multiple experiments were performed. Pre-trained models such as VGG-16 [29] could not be used as its accuracy over the Door dataset was poorer than expected in a pilot test. Hence the decision was made to create a new model architecture and train it from scratch. The experiments performed include adding or removing layers, changing the model hyperparameters, changing the input size based on the model, and changing the optimizer.

The chosen architecture consists of two convolutional layers, two pooling layers, and three fully connected or dense layers. The first convolutional layer consists of

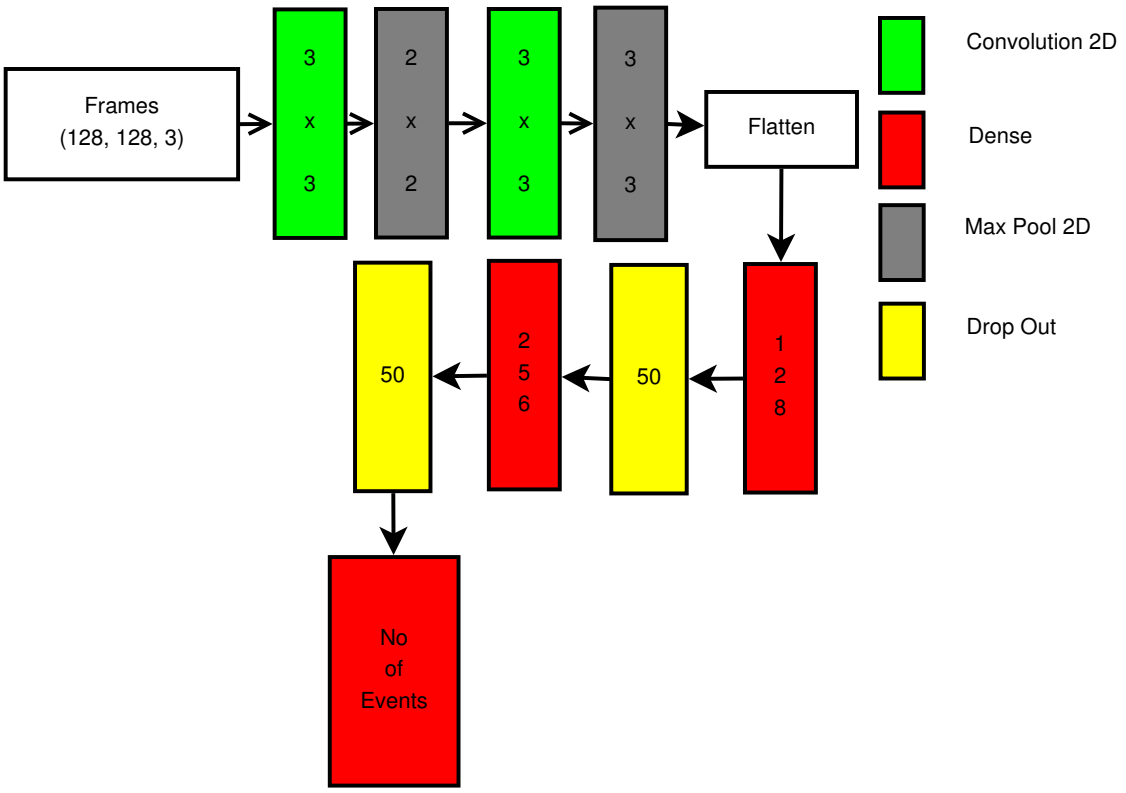


Figure 4.1 *The implemented CNN Architecture*

64 filters and the second convolutional layer consists of 32 filters. The last fully connected layer act as the output layer. Dropout layers (50 %) are used to avoid overfitting. SGD is used as the optimizer. The output is a probability distribution among different events chosen. The units or parameters of the output layer vary based on the dataset used (two events in the case of Door dataset and four in the case of UCF101). A visual representation of the architecture is shown in Figure 4.1.

4.1.2 Training Details

Since CNN accepts only individual images as input, frames belonging to different events are segregated. 70 % of the data is used for training, 20 % is used for validation and, the remaining 10 % is used for testing the model. The same principle is followed for both the datasets, as shown in Table 4.1. The imbalance in the Door dataset (refer to Section 3.1) is handled by allocating equal number of frames for the events. Such balancing is not needed for the UCF101 dataset.

Table 4.1 *Number of Frames per split*

Dataset	Training	Validation	Testing
Door Dataset	7000	2000	1000
UCF101	59340	19800	7989

4.1.3 Input Pre-processing

The image dimensions are represented in the form (height, width, depth). The original image dimensions are (320, 240, 3) and the images are downsampled, i.e. resized to (112, 112, 3), for faster training. Chosen videos are converted into frames and saved.

4.2 CNN-LSTM Model

The idea is to extract characterizing features from frames belonging to each video, using CNN and passing them to LSTM, followed by two fully connected layers, of which the final one performs the prediction. Since the features are extracted from multiple video frames, possibly even all of them, both spatial and temporal information is captured. LSTM handles long term dependencies and hence temporal features are temporal information can be exploited.

4.2.1 CNN-LSTM Architecture

It is possible to use pre-trained deep learning models for the task of feature extraction. The Keras Python module [5] that is used in implementation supports the use of pre-trained models such as VGG (16 and 19 layers) [29], ResNet-50 [17], etc. In this architecture, VGG-16 [29] is chosen to extract features. Though there is no particular reason behind selecting VGG-16 [29], pre-trained VGG is a good feature extractor from images. The pre-trained CNN model is used only for feature extraction from individual frames. The extracted features are merged for each video. However, in the previously implemented CNN model (refer to Section 4.1), the predictions are made for each individual frame rather than entire video.

VGG-16 [29] is trained on ImageNet [9] dataset, consisting of several classes and thousands of images. Hence it will be easier for the pre-trained model to extract characterizing features from the datasets considered.

The frames are passed into the VGG-16 model, which is a python object, and its predict function is used to extract the features frame by frame. The output of the

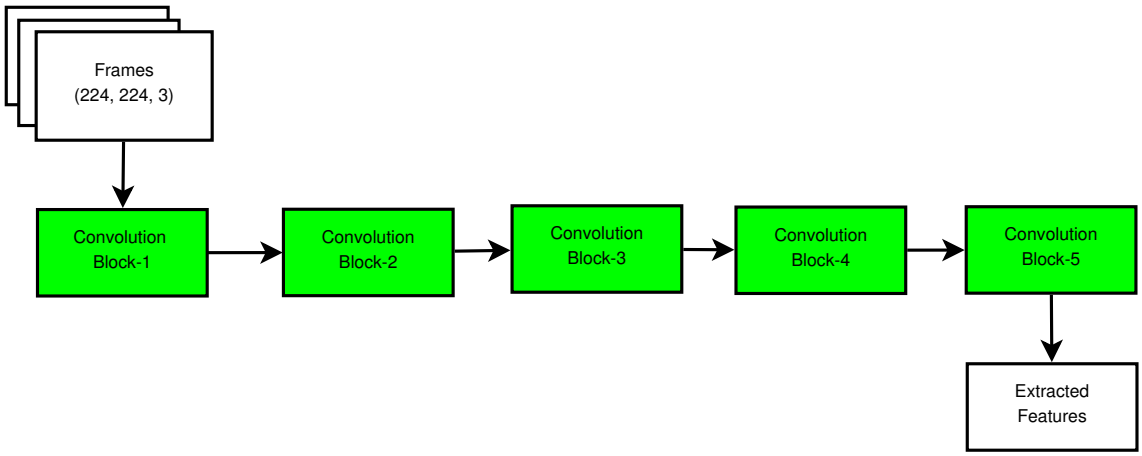


Figure 4.2 *VGG-16 architecture excluding the fully connected layers, is used for feature extraction.*

last convolution block is in the shape of $7 \times 7 \times 512$, which is flattened resulting in a shape 25088. If there are 'X' number of frames in a video, the resulting shape of extracted features is ('X', 25088). This process is repeated over all the videos chosen for training the model. Figure 4.2 depicts the VGG-16 [29] architecture and the layers involved in feature extraction. Convolutional blocks 1 and 2 consists of two convolutional layers and a max pooling layer, while blocks 3, 4, and 5 also include a third convolutional layer. Only the convolution blocks are involved in the feature extraction process.

The extracted features are provided as input to the LSTM layer. The architecture consists of one LSTM layer, and two fully connected layers. In order to avoid overfitting, dropout regularization layers are used. The last fully connected layer with SoftMax activation acts as the output layer. The output is the probability distribution among class labels.

Experiments were performed by adding and removing LSTM layers, dense layers, changing optimizer, and changing the number of neurons per layer. The process was repeated several times to find the optimal architecture. The architectures were evaluated based on its performance on training data, validation data, and the loss values obtained. The performance on test data was used as the ultimate criterion for choosing the best architecture. Figure 4.3 provides a visual representation of the LSTM architecture block diagram.

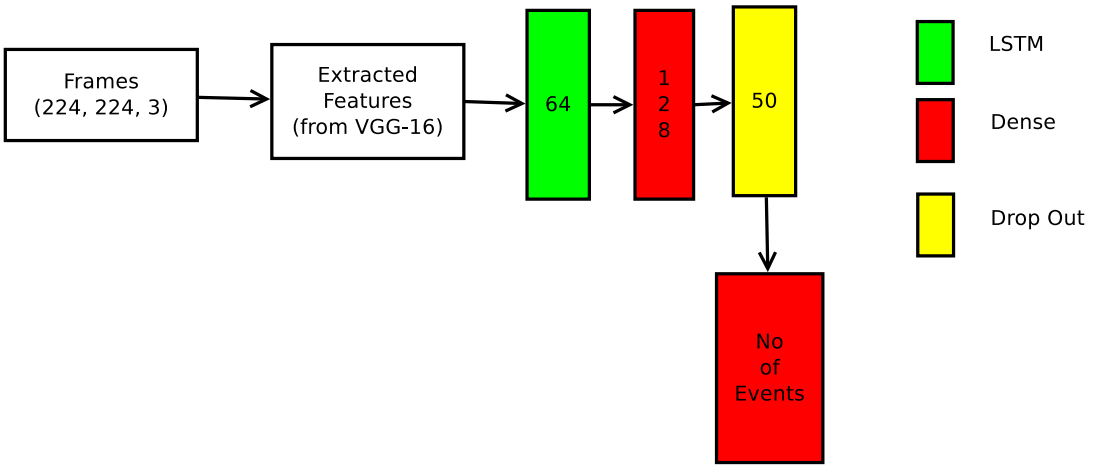


Figure 4.3 *LSTM architecture block diagram.*

4.2.2 Training Details

From the entire dataset, 10 % of videos are used for testing. The rest of the data is split using inbuilt train-test split function of sklearn [26]. 70 % of data is used for training and 30 % is used for validation. The same logic is applied for both datasets. In order to fix the imbalance of the Door dataset (refer to Section 4.1 for additional details), 100 videos belonging to class Entry and 100 videos belonging to class Exit, are used for training the CNN-LSTM model. No such balancing decisions are made on the UCF101 dataset.

4.2.3 Input Pre-processing and Feature Extraction

The image dimensions are represented in the form (height, width, channels). The images are reshaped to following dimensions (224, 224, 3), with 3 channels corresponding to color images. Each reshaped frame is converted into an array and an extra dimension is added to match with the VGG-16 input requirements [29], i.e. (height, width, channels). The entire process is done frame by frame and repeated for each video.

Using pre-trained VGG-16 [29], features are extracted from pre-processed frame arrays. Once feature extraction is completed, the average number of frames in each input video clip is computed. The extracted feature arrays are modified to have uniform shape. If the number of frames is greater than the average, excess frames are removed. On the contrary, if the number of frames is lower than the average, additional frames are added (zero arrays).

After the data manipulation process, the feature arrays with class labels are split into training and validation datasets. The split datasets are trained using LSTM. If the input is in the form of frames, the pre-processing follows above mentioned steps. In the case of input format is video, OpenCV [3] is used to extract frames from the videos. After the frame extraction, pre-processing steps are performed.

4.3 3D-CNN/C3D Model

The idea is to perform 3D convolutions on a set of frames belonging to each video, using a CNN. In order to capture spatial and temporal information, 3D convolutions are performed. The frames are chosen either in random or at specified intervals. The selection of frames is governed mainly by the processing time and the redundant information in frames, e.g. door-only images not featuring any people in the Door dataset. The number of trainable parameters scales (increases) with the number of frames chosen, which impacts the training time and resource consumption. Hence the decision was made to choose 30 frames from each video. The frames are selected in the ways mentioned above.

4.3.1 3D-CNN Architecture

The kernel shape or the dimensions of 3D convolutions is in the form of (height, width, depth). In 3D convolutions, the kernel depth and the image depth are not the same, hence the convolutions are performed in 3 dimensions. This is the difference between 2D and 3D convolutions. Depth helps to retain temporal information across the frames, even after convolutions. Since the input is not a single image but a set of images, the salient characteristics across the images are retained by using 3D convolutions. The architecture consists of two 3D convolutional layers, and one dense layer. Both convolutional layers have 32 filters each. The output layer is a dense layer with SoftMax activation. The output is the probability distribution among class labels. This basis architecture is altered by changing the number of layers, the number of neurons per layer, and the optimization algorithm. Different architectures are evaluated on the basis of training performance, and ultimately the performance on the test data. A visual representation of the architecture is shown in Figure 4.4.

4.3.2 Training Details

10 % of data is used for testing the model. Remaining data is split in the ratio 70:30 where, 70 % is used for training the model and 30 % is used for validation. To

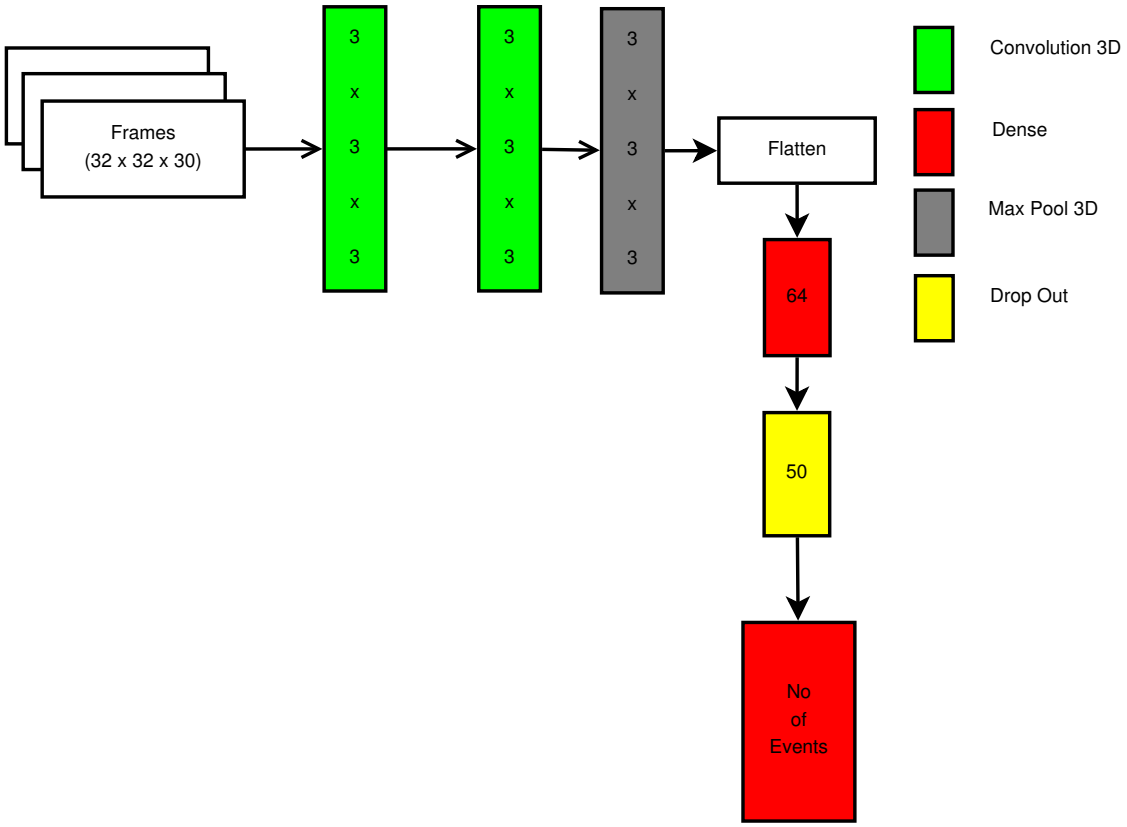


Figure 4.4 *3D-CNN architecture block diagram.*

maintain the balance in Door dataset, almost equal amount of data is used for the Entry and Exit classes. UCF101 [30] dataset does not have a significant imbalance and hence splits can be performed directly.

4.3.3 Input pre-processing

The size of the images (frames) is reduced to dimensions (32, 32, 3). The input image (frame) is a color image. The reason for size reduction is to enable faster processing of the frames and reduce the time taken for model training. Thirty frames are chosen from each video.

Instead of handling the entire set of frames, only a part of it is chosen because of the redundancy in the Door dataset. Instead of manually selecting the frames, a script is written to choose the way in which the frames are selected.

Two approaches are used to choose the frames. The first approach is to choose the frames at regular intervals for e.g. one frame per every five frames, the second approach is to choose the frames randomly without sampling (no repetition). In

either case, the chosen frames are re-sized to (32, 32, 3) to match the input size requirements of the model. This process is repeated for all the videos.

If the input is in the form of frames, the pre-processing steps are done as explained above. If the input format is video, OpenCV [3] is used to extract frames from videos and pre-processing steps are done.

5. PERFORMANCE EVALUATION OF IMPLEMENTED MODELS

In this chapter, the performance of each implemented model in both training and testing phases are discussed. The models are also evaluated on the additional data (refer to Section 3.4).

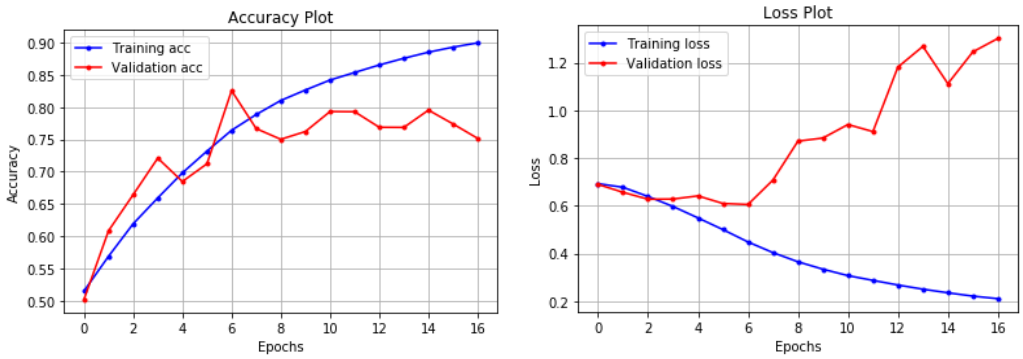
5.1 CNN Model Performance

Once the input pre-processing steps are executed (refer to Section 4.1.1), the model is trained with the input data. The hyperparameters including the number of layers are varied in search of the optimal ones in terms of the chosen performance metrics.

5.1.1 Model Performance — Training

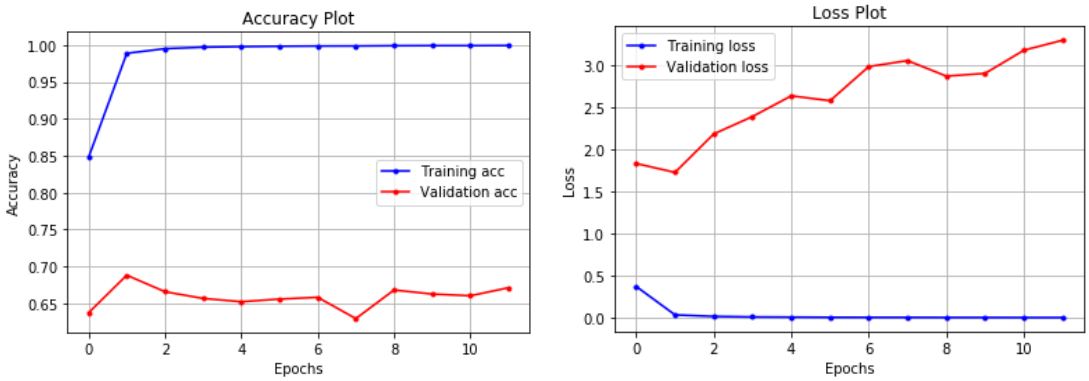
Due to the complexity of deep neural networks, it is important to observe the training process. For example, a poor choice of the hyperparameters may lead into poor performance. Figures 5.1 and 5.2, depicts the performance curves (accuracy and loss) in terms of training and validation splits. The early stopping technique is used to avoid overfitting and hence number of epochs is different for the two datasets. In the case of Door dataset, from the loss curves in Figure 5.1 (b), it is evident that the model overfits. Though the curve is smooth for training loss, the validation loss seems to increase rapidly after initial convergence with training loss. The ever-increasing validation loss indicates that the model is overfitting after sixth epoch.

On the other hand, in the case of UCF101 dataset, there is no convergence between the training and validation loss curves (refer to Figure 5.2 (b)). The validation loss increases rapidly. The accuracy curves in Figure 5.2 (a) suggest that the model is not a good fit for the UCF101 [30] dataset. The model overfits for the given dataset, which can be inferred from the validation loss curve. The validation loss increases as the number of epochs increase.



(a) (b)

Figure 5.1 *CNN—Training Performance Curves for the Door dataset (a) Accuracy versus Epochs (b) Loss versus Epochs*



(a) (b)

Figure 5.2 *CNN—Training Performance Curves for the UCF101 dataset (a) Accuracy versus Epochs (b) Loss versus Epochs*

5.1.2 Model Performance — Evaluation

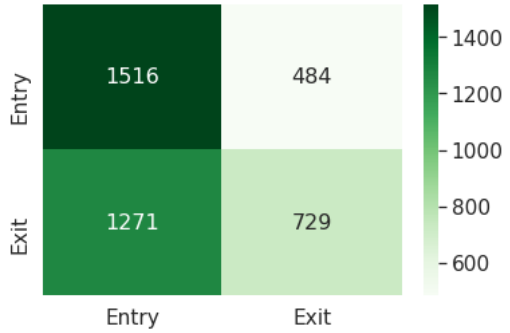
The model is tested with remaining unseen data, i.e. the data that was not used for training or validation. The predictions are made frame-wise. Accuracy is calculated by comparing the ground truth and the prediction made by the model. The number of frames used for testing the model is provided in Table 5.1. The test data includes unused images from the extra Entry videos (refer to Section 3.1 for further details). Accuracy and f1-score for both datasets are shown in the table. Figures 5.3 (a) and (b), depict the confusion matrix.

The lack of temporal data handling, impacts the performance of the model. Even if multiple frames are sent at once to the CNN model, it is not designed to make use of the temporal information.

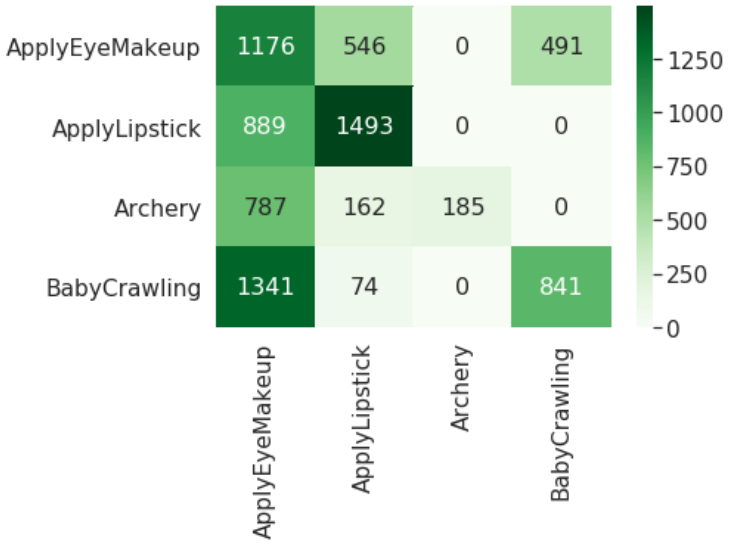
Table 5.1 *Number of Frames per Dataset and Accuracy Metrics*

Dataset	Frames	Accuracy	f1-score
Door Dataset	4000	56	[0.63,0.45]
UCF101	7989	46	[0.37, 0.64, 0.28, 0.46]

Note: *f1-score is represented as an array. For the Door dataset, the array is in the form [Entry, Exit]. In the case of the UCF101 dataset, the array is in the form [Apply Lipstick, Archery, Apply Eye Makeup, Baby Crawling]*



(a)

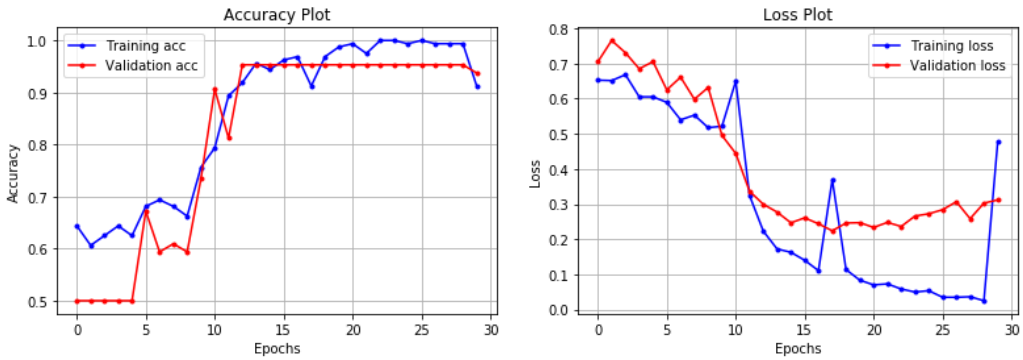


(b)

Figure 5.3 *CNN—Confusion Matrix (a) Door dataset (b) UCF101 dataset*

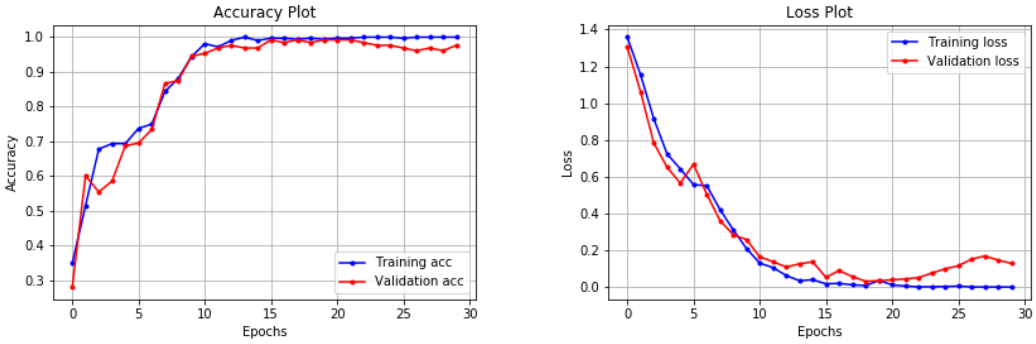
5.2 CNN-LSTM Model Performance

Once the features are extracted, the CNN-LSTM model can be trained. Multiple experiments are conducted in search of optimal hyperparameters.



(a) (b)

Figure 5.4 *CNN-LSTM—Training Performance Curves for the Door dataset (a) Accuracy versus Epochs (b) Loss versus Epochs*

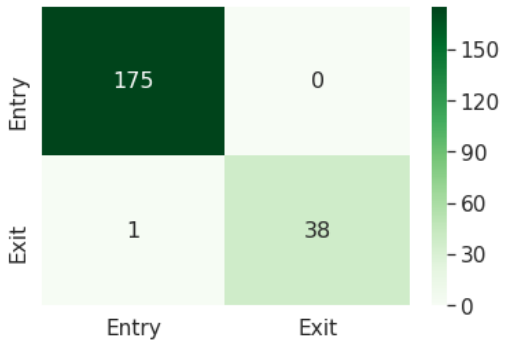


(a) (b)

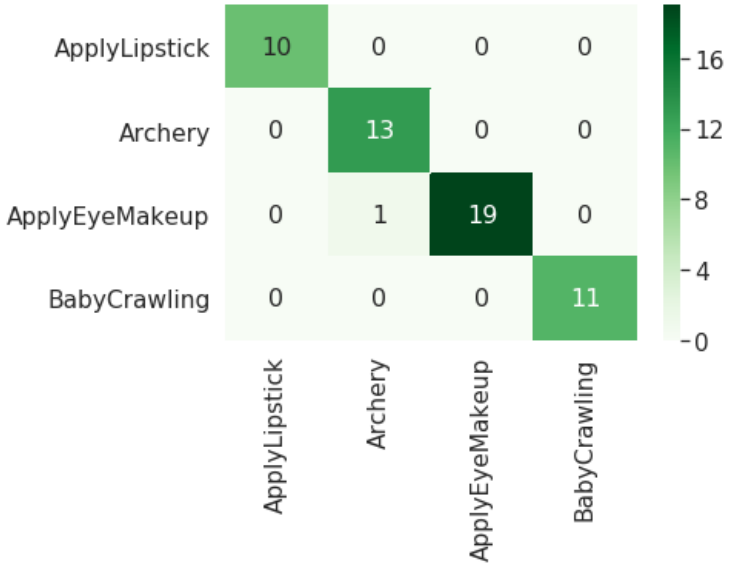
Figure 5.5 *CNN-LSTM—Training Performance Curves for the UCF101 dataset (a) Accuracy versus Epochs (b) Loss versus Epochs*

5.2.1 Model Performance — Training

The training and validation accuracy and loss curves for the Door dataset are shown in Figures 5.4 (a) and (b). The loss curves (training and validation loss) converge only for a short duration, after which the validation loss gradually increases after 16 epochs. Validation accuracy stabilizes after 12 epochs and remains approximately the same for a sustained duration. Figures 5.5 (a) and (b), show the same respective curves for the UCF101 dataset. The loss curves indicate that the loss values are almost the same for both the training and validation splits. However, after 20 epochs, the validation loss gradually increases. The accuracy curves converge and sustain over a period of epochs. Since the training accuracy reaches 1, it is not necessary to train for more epochs. In order to avoid overfitting, training is stopped after 30 epochs, by using the early stopping mechanism.



(a)



(b)

Figure 5.6 *CNN-LSTM—Confusion Matrix (a) Door dataset (b) UCF101 dataset*

5.2.2 Model Performance — Evaluation

The model evaluation is done by testing with unseen data from both datasets. Accuracy is calculated by counting the number of correct predictions made. the confusion matrix and f1-score are also computed (refer to Section 2.6). The model is tested with 55 videos from the UCF101 dataset. In the case of Door dataset, the model is evaluated by testing with 175 videos belonging to the Entry event and 38 videos belonging to the Exit event. The model is evaluated by testing with the entire set of additional data available for Entry event, refer to Section 3.1 for details.

Table 5.2, represents model's accuracy and f1-scores. Figures 5.6 (a) and (b), represents the confusion matrix for both the Door and UCF101 datasets.

Table 5.2 *Number of videos per Dataset and Accuracy Metrics*

Dataset	Videos	Accuracy	f1-score
Door Dataset	214	98.9	[0.99, 0.98]
UCF101	55	98.14	[1, 0.96, 0.97, 1]

Note: *f1-score is represented as an array. For the Door dataset, the array is in the form [Entry, Exit]. In the case of the UCF101 dataset, the array is in the form [Apply Lipstick, Archery, Apply Eye Makeup, Baby Crawling]*

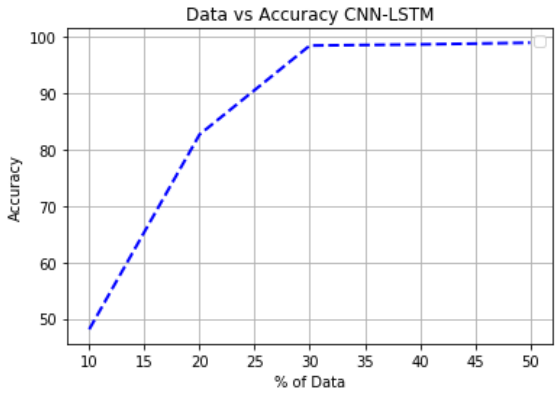


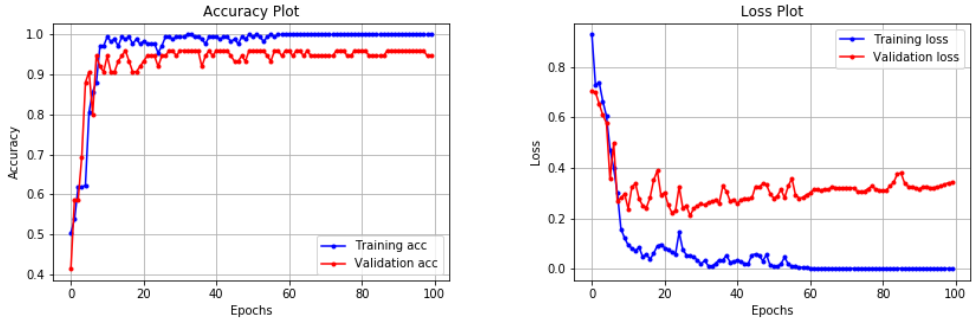
Figure 5.7 *Percentage of Data vs Accuracy*

5.2.3 Amount Of Data Required vs Accuracy

Additional experiments were performed to determine the minimum amount of data required to get the best results. The model training and validation was performed with 10 % of training data. Then evaluation was performed using test data. The experiment was repeated in iterations of 10 % increments in the amount of the data. A clear improvement in accuracy was observed when 30 % of the original dataset was used. Figure 5.7 and Table 5.3 provide information about the amount of data used and the accuracy scores. This process is performed only for CNN-LSTM since it has achieved good results, as presented earlier.

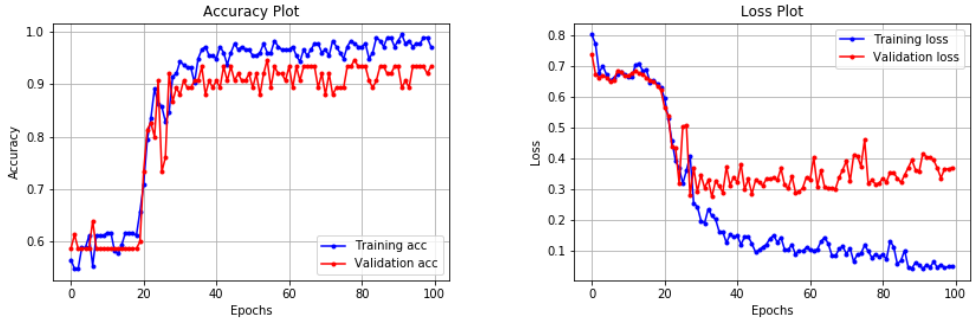
Table 5.3 *Amount of Data used and Accuracy*

Videos	% of Data	Accuracy
20	10	48.13
40	20	82.7
60	30	98.5
80	40	98.7
100	50	98.8



(a) (b)

Figure 5.8 *3D-CNN—Training Performance Curves -Frames from Select Intervals for the Door dataset (a) Accuracy versus Epochs (b) Loss versus Epochs*



(a) (b)

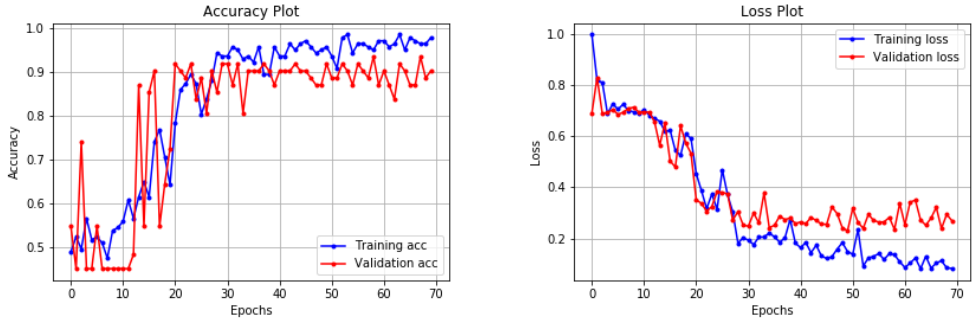
Figure 5.9 *3D-CNN—Training Performance Curves -Frames from Select Intervals for the UCF101 dataset (a) Accuracy versus Epochs (b) Loss versus Epochs*

5.3 3D-CNN Model Performance

Similar to the other two models, experiments are performed to choose the optimal set of model parameters based on the accuracy achieved. Once the architecture is decided, training of the 3D-CNN model is commenced.

5.3.1 Model Performance — Training

Figures 5.8 — 5.10, portray the performance curves (accuracy and loss) for both training and validation. In the CNN and the CNN-LSTM models, all the frames were considered. However, in the 3D-CNN model, 30 frames were selected from each video. Two approaches were followed in frame selection. Random selection of frames was implemented only for Door dataset. The resizing of images to dimensions (32, 32, 3) leads to faster execution.



(a) (b)

Figure 5.10 *3D-CNN—Training Performance Curves -Frames Random Selection for the Door Dataset (a) Accuracy versus Epochs (b) Loss versus Epochs*

Figure 5.8 (b) shows the convergence of the loss value at the initial stages of training, after that the validation loss oscillates for a few epochs and then increases. The training accuracy touches 1 and prolongs over a period of epochs. However, the validation loss is significantly increasing over a period of epochs (after 15 epochs), which indicates that the model is starting to overfit. In the case of the UCF101 dataset, the loss curves converge and sustain over a period of few epochs after which the validation loss increases (refer to Figure 5.9 (b)).

These results for both the datasets involved frames selected at specific intervals, i.e. dropping a fixed number of sequential frames after every chosen frame. Alternatively, frames were selected randomly, without replacement, and sorted in ascending order to retain temporal information.. The validation accuracy of this approach is at times higher than the training accuracy, as seen in Figure 5.10 (a). Though the training and validation loss curves converge in the initial stages, the convergence is sustained (refer to Figure 5.10 (b)) over a period of epochs, after which the validation loss increases.

5.3.2 Model Performance — Evaluation

The model was tested on remaining unseen data. 51 videos from the UCF101 [30] dataset and 213 videos from the Door dataset was used for model testing. There were significantly fewer videos featuring the Exit event than those with the Entry event, so extra videos of the latter class were also employed.

Since training of Door data involved two variants, both variants were evaluated to see which method works the best. Performance metrics include accuracy and f1-score (refer to Table 5.4). The confusion matrices for both datasets are displayed in

Table 5.4 *Number of videos per Dataset and Accuracy Metrics*

Dataset	Frame Selection	Videos	Accuracy	f1-score
Door Dataset	Specific Interval	214	98.13	[0.98, 0.95]
UCF101	Specific Interval	51	77	[0.75, 0.66, 0.79, 0.9]
Door Dataset	Random	214	97.19	[0.95, 0.93]

Note: *f1-score is represented as an array. For the Door dataset, the array is in the form [Entry, Exit]. In the case of the UCF101 dataset, the array is in the form [Apply Lipstick, Archery, Apply Eye Makeup, Baby Crawling]*

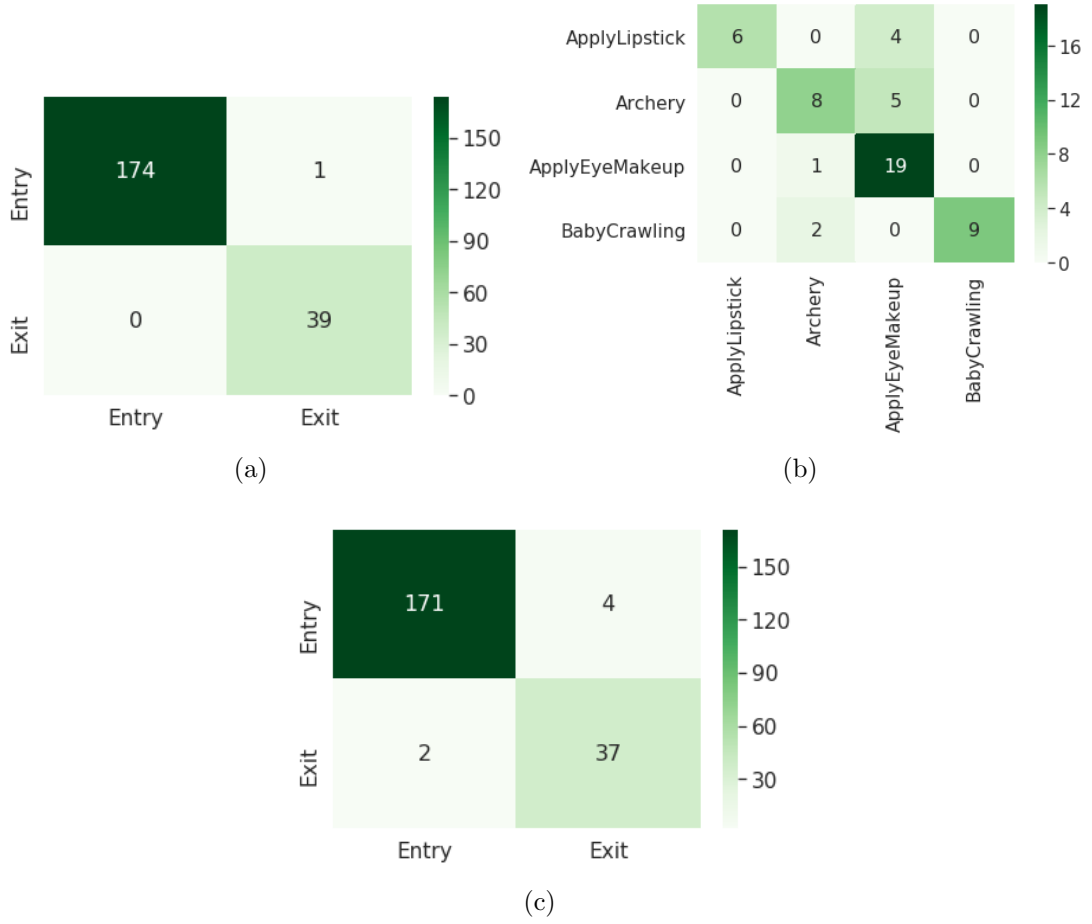


Figure 5.11 *3D-CNN—Confusion Matrix (a) Door dataset (b) UCF101 dataset (c) Door dataset-Random frames*

Figures 5.11 (a), (b), and (c).

5.4 Rationale Behind Model Selection

Additional testing is done only for the CNN-LSTM and 3D-CNN models. The decisions are made purely on the basis of performance. The CNN-LSTM model

achieved good results with the test data for both the UCF101 [30] and the Door dataset (refer to Section 5.2.2). The 3D-CNN, produces good results for the Door dataset, but it is not the same in the case of the UCF101 (refer to Section 5.3.2). The performance of the CNN is relatively low for both for both datasets. Hence the decision was made to perform additional testing for the CNN-LSTM and the 3D-CNN model only. Refer to Table 5.5 for additional details.

Table 5.5 *Models on which additional testing is done*

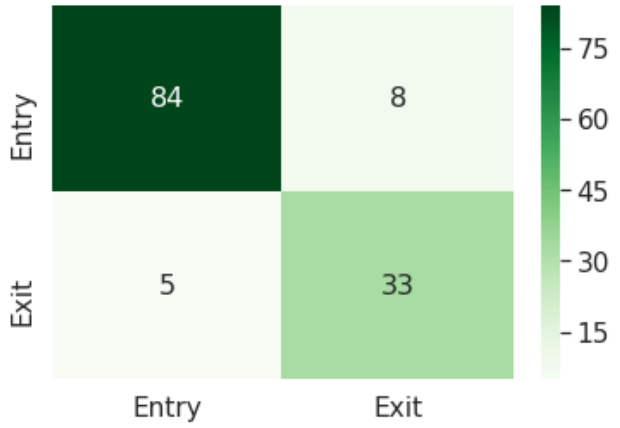
Model	Door dataset	UCF101 dataset
CNN-LSTM	Yes	Yes
3D-CNN	Yes	No
CNN	No	No

5.4.1 Additional Testing — CNN-LSTM

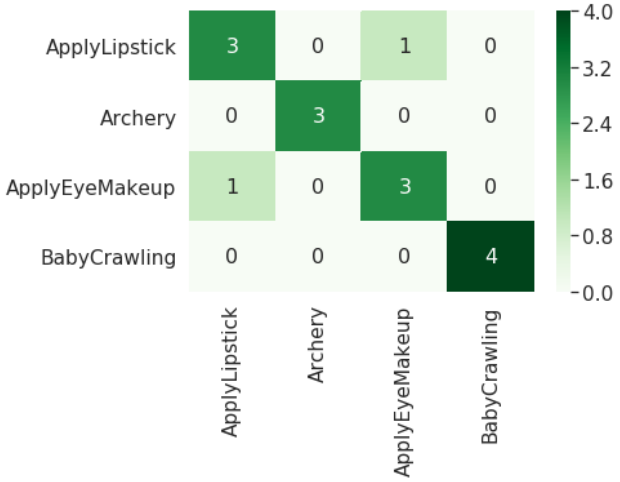
Accuracy and f1-score can be interpreted from Table 5.6, for both datasets.The videos from additional UCF101 data (Section 3.4) are generally longer (approximately 60-70 seconds). Accordingly, more frames were examined for such videos than for training ones. There were two incorrect classifications out of 15 videos evaluated in the UCF additional data, which corresponds to an accuracy of 87 %. In the case of the Door dataset, there are 13 incorrect classifications out of 130 videos tested, which corresponds to an accuracy of 90 %. A visual description about incorrect classifications can be seen using the confusion matrix displayed in Figure 5.12 (a) and (b).

5.4.2 Additional testing — 3D-CNN

Accuracy and f1-score metrics are shown in Table 5.7. Since there are two approaches used in the frame selection process, both the approaches are evaluated. In the case of frames selected from specific intervals, there were 15 incorrect classifications out of 130 videos tested, which corresponds to an accuracy of 88.4 %. On the other hand, frames selected in random process yields 19 incorrect classifications, which corresponds to accuracy of 85 %. The confusion matrices for both the cases are shown in Figure 5.13 (a) and (b).



(a)



(b)

Figure 5.12 *CNN-LSTM—Additional Data Confusion Matrix (a) Door dataset (b) UCF101 dataset*

Table 5.6 *Additional videos per Dataset and Accuracy Metrics*

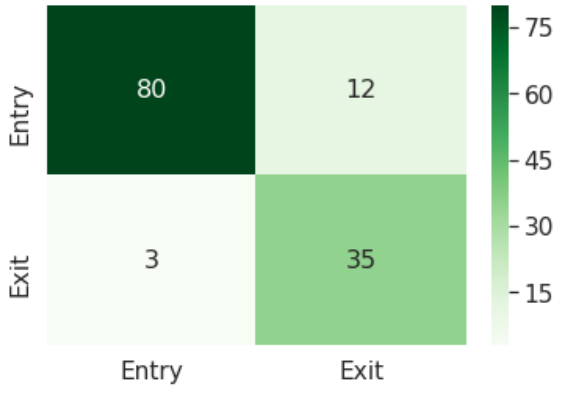
Dataset	Videos	Accuracy	f1-score
Door Dataset	130	90	[0.92, 0.83]
UCF101	15	87	[0.75, 1, 0.75, 1]

Note: *f1-Score is represented as an array. For the Door dataset, the array is in the form [Entry, Exit]. In case of the UCF101 dataset, the array is in the form [Apply Lipstick, Archery, Apply Eye Makeup, Baby Crawling]*

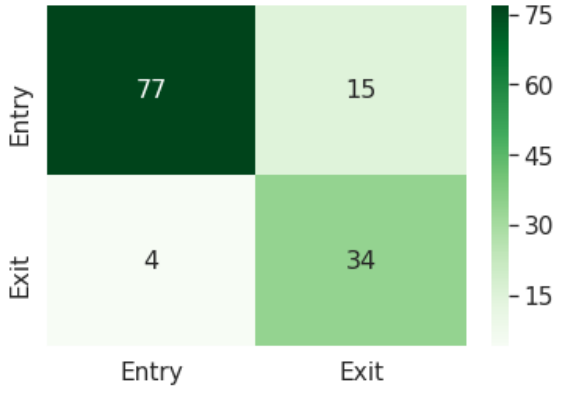
Table 5.7 *Door Dataset Video Count and Accuracy Metrics*

Frame Selection	Videos	Accuracy	f1-score
Specific Interval	130	88.46	[0.91, 0.82]
Random	130	85	[0.89, 0.78]

Note: *f1-Score is represented an array, [Entry, Exit].*



(a)



(b)

Figure 5.13 *3D-CNN —Additional Data—Confusion Matrix (a) Door dataset(Frames-Specific intervals) (b) Door dataset(Frames- Random selection)*

6. INTERPRETATIONS AND INSIGHTS

This chapter discusses the performance primarily based on the Door dataset, since it is the custom dataset and the main focus of the thesis. Analysis is done on the performance of each model and conclusions are derived.

6.1 Performance of CNN Model

CNNs are used to perform image classification with good success [29], their ability is tested since videos consists of individual images. To put it in a perspective, the videos are converted into frames and image classification is performed on the frames. However, the performance is significantly lower in terms of accuracy and f1-score when compared with the other two approaches. This is likely due to the models inability to exploit the temporal dependencies of the data.

Though spatial features are extracted, temporal information plays a vital role in event classification. As shown by the obtained results, the CNN approach produces poor results. Especially in the Door dataset there are similarities between both the events, which further impacts the performance. Training of the CNN-based model is also time consuming, due to the volume of images and limited computing resources. The CNN approach can be used as a benchmark for comparison, but it is not suitable for the event classification problem targeted in this thesis.

6.2 Performance of CNN-LSTM Model

The CNN-LSTM approach, takes advantage of using a pre-trained model to extract characterizing features. The process of extracting features from frames can be time consuming, but it is only performed once. The extracted features can be saved and various experiments can be performed on it. However, the training time of the CNN-LSTM model is relatively low compared to the CNN approach. For the limited resources available, the CNN-LSTM approach is the ideal is the ideal solution for the event classification task targeted in this thesis.

Figure 6.1 *An example of an incomplete Entry event. The door is left open.*

Figure 6.2 *An example of an incomplete Entry event, where a third person closes the door.*

The model is able to classify the events even in the presence of multiple people doing the same action. However, the model fails to recognize the event if it is left incomplete. Since the occurrence of the incomplete events is relatively rare, their impact is negligible. Some of the cases where the model fails to identify the events include incomplete events, e.g. when the door is left open after the events occurrence. Such occasions are demonstrated in Figures 6.1 and 6.2. One of the scenarios requiring further concern is deciding which event is occurring in the cases that are regarded neither as Entry nor as Exit. For example, if both Entry and Exit events occur simultaneously, what is the expected classification? Multilabel classification could be a potential solution for this case, though such videos were not present in the dataset. However, by thresholding the probability, both classes can be rejected if neither of the class probabilities reaches the threshold. This could be used as a potential workaround for cases that are neither Entries nor Exits.

However, with additional data and appropriate labeling, the model can be trained to recognize incomplete events. The number of frames taken into consideration (currently set to the average number of frames) also plays an important role. If the value is set too high or to the values based on longer videos, the accuracy decreases. Hence it is better to keep it in the proximity of the average value.

6.3 Performance of 3D-CNN Model

The 3D-CNN approach is also efficient in terms of capturing both spatial and temporal features. It produces good results with test data, although the performance decreases when the number of classes is high (UCF dataset). For the Door dataset, it performs similar to CNN-LSTM. However, the CNN-LSTM approach is able to adapt to UCF101, while the 3D-CNN approach struggles.

The image size is reduced to (32, 32, 3), which drastically speeds up the training process by decreasing the number of trainable parameters. A higher number of epochs can be run in reasonable time. Another area of concern is the the selection of frames for prediction, which will impact the performance. In the case of random frame selection while retaining temporal order, performance might vary every time the evaluation script is run. The 3D-CNN model, struggles to the predict the incomplete events and the length of the video might impact the prediction.

7. CONCLUSION

The objective of this thesis was to identify a suitable model for the event classification task. Three models namely CNN, CNN-LSTM, and 3D-CNN were implemented. With each model, a different approach was implemented to perform the task. Accuracy and f1-scores were used as the key performance indicators. The CNN model implemented in this thesis, achieved 56 % accuracy, 3D-CNN achieved 98.13 % accuracy and CNN-LSTM model achieved 98.9 % on test dataset i.e., for data not used in the training process. Comparing the performance metrics of the implemented models, it is evident that the CNN-LSTM model produces the best results. Since a pre-trained model is used for feature extraction, a significant amount of time is saved in designing the rest of the model architecture model architecture.

The 3D-CNN model produces results similar to the CNN-LSTM model. However, not all the frames from a video are used to train the model. The process of frame selection from a video is performed is performed in two ways, either choosing the frames at regular intervals or in a random manner. The frames are sorted so that the temporal information is preserved. The training time is lower compared to other implemented models, since no explicit pre-processing or feature extraction is performed. However, with additional testing, CNN-LSTM model outperforms 3D-CNN model. If the frames are chosen randomly, the results are highly dependent on the selected frames. These factors contribute to choosing CNN-LSTM as the best model for the tested datasets. The implemented CNN model is not able to exploit the temporal dependency of the data. However, CNNs have been successfully used for the image classification task. Therefore the CNN model was used as a benchmark for the other models.

Future work includes training the CNN-LSTM model with additional data. This includes training the model with data captured from different camera positions and using videos belonging to different qualities. Another challenge is making real-time predictions. Testing the model with more difficult cases such as simultaneous entry and exit, multiple people performing different events at the same instant is also considered in the pipeline for future work.

BIBLIOGRAPHY

[1] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. I. Verkamo, "Advances in knowledge discovery and data mining," U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, Eds. Menlo Park, CA, USA: American Association for Artificial Intelligence, 1996, pp. 307–328. [Online]. Available: http://dl.acm.org/citation.cfm?id=257938.257975

[2] D. Arthur and S. Vassilvitskii, "K-means++: The advantages of careful seeding," in *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics, 2007, pp. 1027–1035. [Online]. Available: http://dl.acm.org/citation.cfm?id=1283383.1283494

[3] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[4] O. Chapelle, P. Haffner, and V. N. Vapnik, "Support vector machines for histogram-based image classification," *Trans. Neur. Netw.*, vol. 10, no. 5, pp. 1055–1064, Sep. 1999. [Online]. Available: https://doi.org/10.1109/72.788646

[5] F. Chollet *et al.*, "Keras," https://keras.io, 2015, accessed: 2019-03-31.

[6] F. Chollet, *Deep Learning with Python*, 1st ed. Greenwich, CT, USA: Manning Publications Co., 2017.

[7] C. Colah, "Understanding lstm networks," http://colah.github.io/posts/2015-08-Understanding-LSTMs/, accessed: 2019-03-31.

[8] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, Sep. 1995. [Online]. Available: https://doi.org/10.1023/A:1022627411411

[9] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR*, 2009.

[10] L. Deng and X. Li, "Machine learning paradigms for speech recognition: An overview," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 5, pp. 1060–1089, May 2013.

[11] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, "Long-term recurrent convolutional networks for visual recognition and description," in *CVPR*, 2015.

[12] H. Eom and H. Choi, "Alpha-pooling for convolutional neural networks," *CoRR*, vol. abs/1811.03436, 2018. [Online]. Available: http://arxiv.org/abs/1811.03436

[13] C. Feichtenhofer, A. Pinz, and A. Zisserman, "Convolutional two-stream network fusion for video action recognition," *CoRR*, vol. abs/1604.06573, 2016. [Online]. Available: http://arxiv.org/abs/1604.06573

[14] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[15] C. Goutte and E. Gaussier, "A probabilistic interpretation of precision, recall and f-score, with implication for evaluation," in *Proceedings of the 27th European Conference on Advances in Information Retrieval Research*. Springer-Verlag, 2005, pp. 345–359. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-31865-1_25

[16] S. S. Haykin, *Neural networks and learning machines*, 3rd ed. Upper Saddle River, NJ: Pearson Education, 2009.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[18] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997. [Online]. Available: http://dx.doi.org/10.1162/neco.1997.9.8.1735

[19] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects," *Science*, vol. 349, no. 6245, pp. 255–260, 2015. [Online]. Available: http://science.sciencemag.org/content/349/6245/255

[20] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *CVPR*, 2014.

[21] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*. USA: Curran Associates Inc., 2012, pp. 1097–1105. [Online]. Available: http://dl.acm.org/citation.cfm?id=2999134.2999257

[22] Y. LeCun, P. Haffner, L. Bottou, and Y. Bengio, *Object Recognition with Gradient-Based Learning*. Berlin, Heidelberg: Springer Berlin

Heidelberg, 1999, pp. 319–345. [Online]. Available: https://doi.org/10.1007/3-540-46805-6_19

[23] T. M. Mitchell, *Machine Learning*, 1st ed. New York, NY, USA: McGraw-Hill, Inc., 1997.

[24] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on International Conference on Machine Learning*. USA: Omnipress, 2010, pp. 807–814. [Online]. Available: http://dl.acm.org/citation.cfm?id=3104322.3104425

[25] J. Y. Ng, M. J. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, "Beyond short snippets: Deep networks for video classification," *CoRR*, vol. abs/1503.08909, 2015. [Online]. Available: http://arxiv.org/abs/1503.08909

[26] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[27] S. Ruder, "An overview of gradient descent optimization algorithms," *CoRR*, vol. abs/1609.04747, 2016. [Online]. Available: http://arxiv.org/abs/1609.04747

[28] A. C. Sarah G., *Introduction to Machine Learning with Python*, 1st ed. USA: O'Reilly Media, Inc., 2016.

[29] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[30] K. Soomro, A. R. Zamir, and M. Shah, "UCF101: A dataset of 101 human actions classes from videos in the wild," *CoRR*, vol. abs/1212.0402, 2012. [Online]. Available: http://arxiv.org/abs/1212.0402

[31] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, Jan. 2014. [Online]. Available: http://dl.acm.org/citation.cfm?id=2627435.2670313

[32] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*, 1st ed. Cambridge, MA, USA: MIT Press, 1998.

[33] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *CoRR*, vol. abs/1409.4842, 2014. [Online]. Available: http://arxiv.org/abs/1409.4842

[34] Y. Tang, "Deep learning using support vector machines," *CoRR*, vol. abs/1306.0239, 2013. [Online]. Available: http://arxiv.org/abs/1306.0239

[35] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," in *ICCV*, 2015.

[36] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool, "Temporal segment networks for action recognition in videos," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2018.

[37] B. Xu, N. Wang, T. Chen, and M. Li, "Empirical evaluation of rectified activations in convolutional network," *CoRR*, vol. abs/1505.00853, 2015. [Online]. Available: http://arxiv.org/abs/1505.00853

[38] M. D. Zeiler and R. Fergus, "Stochastic pooling for regularization of deep convolutional neural networks," *CoRR*, vol. abs/1301.3557, 2013. [Online]. Available: https://arxiv.org/pdf/1301.3557.pdf

[39] X. Zhu, "Semi-supervised learning literature survey," Computer Sciences, University of Wisconsin-Madison, Tech. Rep. 1530, 2005. [Online]. Available: http://pages.cs.wisc.edu/~jerryzhu/pub/ssl_survey.pdf